

# SPEC-CCP-RS

Last changed:	20.11.2025 09:23:52
Version:	3.0.0-rc.6
Creator:	VDV ETS

## Table of Contents

1	Introduction	9
2	Customer Contract Partner System	9
2.1	Customer Contract Partner System	12
2.2	Customer Contract Partner Back-Office Main Module	12
2.3	Customer Contract Partner Back-Office Action Ordering Module	13
2.4	Customer Contract Partner Back-Office Order Execution Module	13
2.5	Customer Contract Partner Back-Office Static Entitlement Module	13
3	Notification Process Patterns	13
3.1	Supporting activities	16
3.1.1	Perform transaction to XY	16
3.1.2	Prepare XY parameters	19
3.2	Supporting classes	20
3.2.1	Action parameters	20
3.2.2	XY parameters	20
3.2.3	SignedXYAttestation	20
3.2.4	Notification parameters	20
3.2.5	XYNotification	21
3.2.6	ForwardedXYNotification	21
3.2.7	tNotifyXY	21
3.2.8	tNotifyXYAborted	21
3.2.9	notifyXY	21
3.2.10	notifyXYResponse	21
3.2.11	notifyXYException	21
3.2.12	forwardXYNotification	21
3.2.13	forwardXYNotificationResponse	21
3.2.14	forwardXYNotificationException	21
3.3	Perform entitlement XY and notify	21
3.3.1	Perform entitlement XY and notify	22
3.3.2	T-Module::Perform entitlement XY and notify	23
3.3.3	Notify entitlement XY	24
3.3.4	Notify entitlement XY aborted	25
3.3.5	Notify entitlement XY aborted based on attestation	25
3.3.6	Notify XY (entitlement owned)	26
3.3.7	Notify XY (entitlement non-owned)	27
3.4	Perform application XY and notify	27
3.4.1	Perform application XY and notify	28
3.4.2	T-Module::Perform application XY and notify	28
3.4.3	Notify application XY	30

3.4.4	Notify application XY aborted	30
3.4.5	Notify XY (application owned)	31
3.4.6	Notify XY (application non-owned)	32
3.5	Handle entitlement XY notification from operational perspective	32
3.5.1	Handle entitlement XY notification from operational perspective	33
3.5.2	Role-BO-Module::Handle entitlement XY notification from operational perspective	33
3.5.3	Check and process entitlement XY notification from operational perspective	34
3.5.4	Role-BO-Module::Handle entitlement XY aborted notification from operational perspective	35
3.6	Handle application XY notification from operational perspective	36
3.6.1	Handle application XY notification from operational perspective	37
3.6.2	Role-BO-Module::Handle application XY notification from operational perspective	37
3.6.3	Check and process application XY notification from operational perspective	38
3.6.4	Role-BO-Module::Handle application XY aborted notification from operation perspective	40
3.7	Handle entitlement XY notification from product perspective	40
3.7.1	Handle entitlement XY notification from product perspective	41
3.7.2	PO-BO-Module::Handle entitlement XY notification from product perspective	41
3.7.3	Check and process entitlement XY notification from product perspective	43
3.7.4	Conditionally forward notification to pCCP	43
3.8	Handle entitlement XY notification from contractual perspective	44
3.8.1	Handle entitlement XY notification from contractual perspective	45
3.8.2	pCCP-BO-Module::Handle entitlement XY notification from contractual perspective	45
3.8.3	Check and process entitlement XY notification from contractual perspective	46
3.8.4	pCCP-BO-Module::Perform specific contractual tasks on entitlement XY notification	47
3.9	Handle application XY notification from contractual perspective	48
3.9.1	Handle application XY notification from contractual perspective	49
3.9.2	pCCP-BO-Module::Handle application XY notification from contractual perspective	49
3.9.3	Check and process application XY notification from contractual perspective	50
3.9.4	pCCP-BO-Module::Perform specific contractual tasks on application XY notification	51
4	Verifying Lists Updated via Increments	52
4.1	Checksum calculation for hotlist and action list verification	54
4.2	Example calculation for an action list inventory	54
4.3	Example calculation for an application hotlist inventory	55
4.4	Example calculation for an entitlement hotlist inventory	56
5	Basic Bundle Back-Office System	57
5.1	Overview	57
5.2	Use Cases	58
5.2.1	Handle discarded messages information	58
5.2.2	Set service as available for a participant	60

5.2.3	Set service as unavailable for a participant	62
5.2.4	Retrieve CV certificate over signing key	64
5.2.5	Retrieve and distribute the CA certificate repository	66
5.2.6	Retrieve and distribute the CV certificate revocation list	68
5.2.7	Notify events	70
5.2.8	Handle events notification	72
5.2.9	Demand application hotlisting	74
5.2.10	Determine user medium owner	76
5.2.11	Demand entitlement hotlisting	78
5.2.12	Demand SAM hotlisting	80
5.2.13	Determine SAM owner	82
5.2.14	Revoke application hotlisting demand	84
5.2.15	Revoke entitlement hotlisting demand	86
5.2.16	Retrieve entitlement hotlist	88
5.2.17	Optional: Retrieve entitlement hotlist with product information	90
5.2.18	Optional: Retrieve incremental entitlement hotlist	92
5.2.19	Optional: Verify entitlement hotlist updated via increments	94
5.2.20	Update organisation hotlist inventory	96
5.2.21	Update SAM hotlist inventory	98
5.2.22	Retrieve and distribute organisation list	100
6	Basic Bundle Terminal Operator System - Foundation	102
6.1	Overview	102
6.2	Use Cases	102
6.2.1	Handle SAM hotlisting demand	102
6.2.2	Check and add SAM to hotlist	105
6.2.3	Retrieve application hotlist	107
6.2.4	Optional: Retrieve incremental application hotlist	109
6.2.5	Optional: Verify application hotlist updated via increments	111
6.2.6	Update authentication key hotlist inventory	113
6.2.7	Update hotlist inventory from operational perspective	115
6.2.8	Distribute SAM configuration	117
6.2.9	Optional: Distribute the SAM reset script	119
6.2.10	Distribute tariff module	121
6.2.11	Monitor SAMs from operational perspective	123
7	Basic Bundle Terminal Operator System - Extended Logging	125
7.1	Overview	125
7.2	Use Cases	125
7.2.1	Optional: Process extended logging for an entitlement	125
7.2.2	Optional: Process extended logging for an application	127
8	Basic Bundle Terminal Operator System - UM with Application	129
8.1	Overview	129
8.2	Use Cases	129

8.2.1	Handle defective user medium with application	129
8.2.2	Optional: Determine UM app instance ID for Medium ID	131
8.2.3	Handle application blocked notification from operational perspective	133
8.2.4	Handle entitlement blocked notification from operational perspective	135
9	Basic Bundle CCP-System - Foundation	137
9.1	Overview	137
9.2	Use Cases	137
9.2.1	Handle application hotlisting demand	137
9.2.2	Handle revocation for application hotlisting demand	140
9.2.3	Handle entitlement hotlisting demand	142
9.2.4	Handle revocation for entitlement hotlisting demand	144
9.2.5	Analyse entitlement history from contractual perspective	146
9.2.6	Resolve notifications with timeout warnings	148
10	Basic Bundle CCP-System - UM with Application	150
10.1	Overview	150
10.2	Use Cases	150
10.2.1	Handle application blocked notification from contractual perspective	150
10.2.2	Handle entitlement blocked notification from contractual perspective	152
10.2.3	Analyse application history from contractual perspective	154
10.2.4	Check entitlement notifications against issuance notification from contractual perspective	156
11	Basic Bundle CCP-System - UM in Customer Center	158
11.1	Overview	158
11.2	Use Cases	158
11.2.1	Handle application unblocked notification from operational perspective	158
11.2.2	Handle application unblocked notification from contractual perspective	161
11.2.3	Handle entitlement unblocked notification from operational perspective	163
11.2.4	Handle entitlement unblocked notification from contractual perspective	165
11.2.5	Optional: Handle lost user medium with application	167
11.2.6	Optional: Handle application terminated notification from operational perspective	169
11.2.7	Optional: Handle application terminated notification from contractual perspective	171
12	Basic Bundle CCP-System - UM with Customer Data	173
12.1	Overview	173
12.2	Use Cases	173
12.2.1	Personalise application	173
12.2.2	Process new information about customer and discounts	175
13	Basic Bundle CCP-System - Extension for UM with Application	177
13.1	Overview	177
13.2	Use Cases	177
13.2.1	Process media shipment list	177

14	Electronic Ticket Bundle CCP-System	179
14.1	Overview	179
14.2	Use Cases	179
14.2.1	Handle entitlement issued notification from operational perspective	179
14.2.2	Handle entitlement issued notification from contractual perspective	181
14.2.3	Handle entitlement terminated notification from operational perspective	183
14.2.4	Handle entitlement terminated notification from contractual perspective	185
14.2.5	Handle entitlement inspected notification from contractual perspective	187
15	Account-Based Payment Bundle CCP-System	189
15.1	Overview	189
15.2	Use Cases	189
15.2.1	Handle entitlement issued notification from operational perspective	189
15.2.2	Handle entitlement issued notification from contractual perspective	191
15.2.3	Handle entitlement terminated notification from operational perspective	193
15.2.4	Handle entitlement terminated notification from contractual perspective	195
16	Stored-Value Payment Bundle CCP-System	197
16.1	Overview	197
16.2	Use Cases	197
16.2.1	Handle entitlement issued notification from operational perspective	197
16.2.2	Handle entitlement issued notification from contractual perspective	200
16.2.3	Handle entitlement terminated notification from operational perspective	202
16.2.4	Handle entitlement terminated notification from contractual perspective	204
16.2.5	Handle stored-value payment method recharged notification from operational perspective	206
16.2.6	Handle stored-value payment method recharged notification from contractual perspective	208
16.2.7	Handle stored-value payment method reimbursed notification from operational perspective	210
16.2.8	Handle stored-value payment method reimbursed notification from contractual perspective	212
17	Sale Electronic Ticket via Account-Based Payment Bundle CCP-System	214
17.1	Overview	214
17.2	Use Cases	214
17.2.1	Handle account-based payment method debited notification from operational perspective	214
17.2.2	Handle account-based payment method debited notification from contractual perspective	217
17.2.3	Handle account-based payment method credited notification from operational perspective	219
17.2.4	Handle account-based payment method credited notification from contractual perspective	221
18	Sale Electronic Ticket via Stored-Value Payment Bundle CCP-System	223
18.1	Overview	223

18.2	Use Cases	223
18.2.1	Handle stored-value payment method debited notification from operational perspective	223
18.2.2	Handle stored-value payment method debited notification from contractual perspective	225
18.2.3	Handle stored-value payment method credited notification from operational perspective	227
18.2.4	Handle stored-value payment method credited notification from contractual perspective	229
19	IN-OUT Bundle CCP-System	231
19.1	Overview	231
19.2	Use Cases	231
19.2.1	Handle check-in notification from contractual perspective	231
19.2.2	Handle check-out notification from contractual perspective	234
19.2.3	Optional: Handle user tariff parameters changed notification from operational perspective	236
19.2.4	Optional: Handle user tariff parameters changed notification from contractual perspective	238
19.2.5	Optional: Handle account-based payment method charging from contractual perspective	240
20	Ordered Action Management Bundle Ordering-CCP-System	242
20.1	Overview	242
20.2	Use Cases	242
20.2.1	Order entitlement issuance	242
20.2.2	Order entitlement unblocking	245
20.2.3	Order entitlement termination	246
20.2.4	Order entitlement blocking	248
20.2.5	Order group	250
20.2.6	Cancel order	252
20.2.7	Handle order obsolescence notification	254
20.2.8	Handle ordered entitlement issued notification from contractual perspective	256
20.2.9	Handle ordered entitlement unblocked notification from contractual perspective	258
20.2.10	Handle ordered entitlement terminated notification from contractual perspective	260
20.2.11	Handle ordered entitlement blocked notification from contractual perspective	262
20.2.12	Analyse order history from contractual perspective	264
21	Ordered Action Management Bundle Executing-CCP-System	266
21.1	Overview	266
21.2	Use Cases	266
21.2.1	Handle ordered entitlement issued notification from operational perspective	266
21.2.2	Handle ordered entitlement unblocked notification from operational perspective	269
21.2.3	Handle ordered entitlement terminated notification from operational perspective	271
21.2.4	Optional: Handle ordered entitlement blocked notification from operational perspective	273

21.2.5	Process action list retrieval configuration	275
21.2.6	Retrieve action list	277
21.2.7	Optional: Retrieve incremental action list	279
21.2.8	Optional: Verify action list updated via increments	281
21.2.9	Update action list inventory from operational perspective	283
22	Static Entitlements Bundle CCP-System	285
22.1	Overview	285
22.2	Use Cases	285
22.2.1	Handle static entitlement issued notification from operational perspective	285
22.2.2	Handle static entitlement issued notification from contractual perspective	288
22.2.3	Handle static entitlement inspected notification from contractual perspective	290
22.2.4	Handle static entitlement terminated notification from operational perspective	292
22.2.5	Handle static entitlement terminated notification from contractual perspective	294
22.2.6	Check static entitlement notifications against issuance notification from contractual perspective	296
23	Miscellaneous Bundle CCP-System	298
23.1	Overview	298
23.2	Use Cases	298
23.2.1	Optional: Retrieve valid entitlements for given app instance ID	298
23.2.2	Optional: Trigger entitlement issuance	301
23.2.3	Optional: Handle entitlement validated notification from contractual perspective	303



# 1 Introduction

From an EFM perspective, the customer contract partner system manages customers, their entitlements and applications. It monitors and checks ongoing contractual aspects with regard to incoming notifications.

This reference specification contains all functionality bundles and use cases for a customer contract partner back-office system.

Depending on the deployment variant, different functionality bundles can be combined.

For a fully functional customer contract partner back-office system, all basic packages must be implemented. There are separate specifications for customer contract partner back-office systems with limited functionality.

# 2 Customer Contract Partner System

This chapter contains all components and interfaces that are involved with the [Customer Contract Partner System](#). Depending on the deployment variant, not all components and interfaces are required.

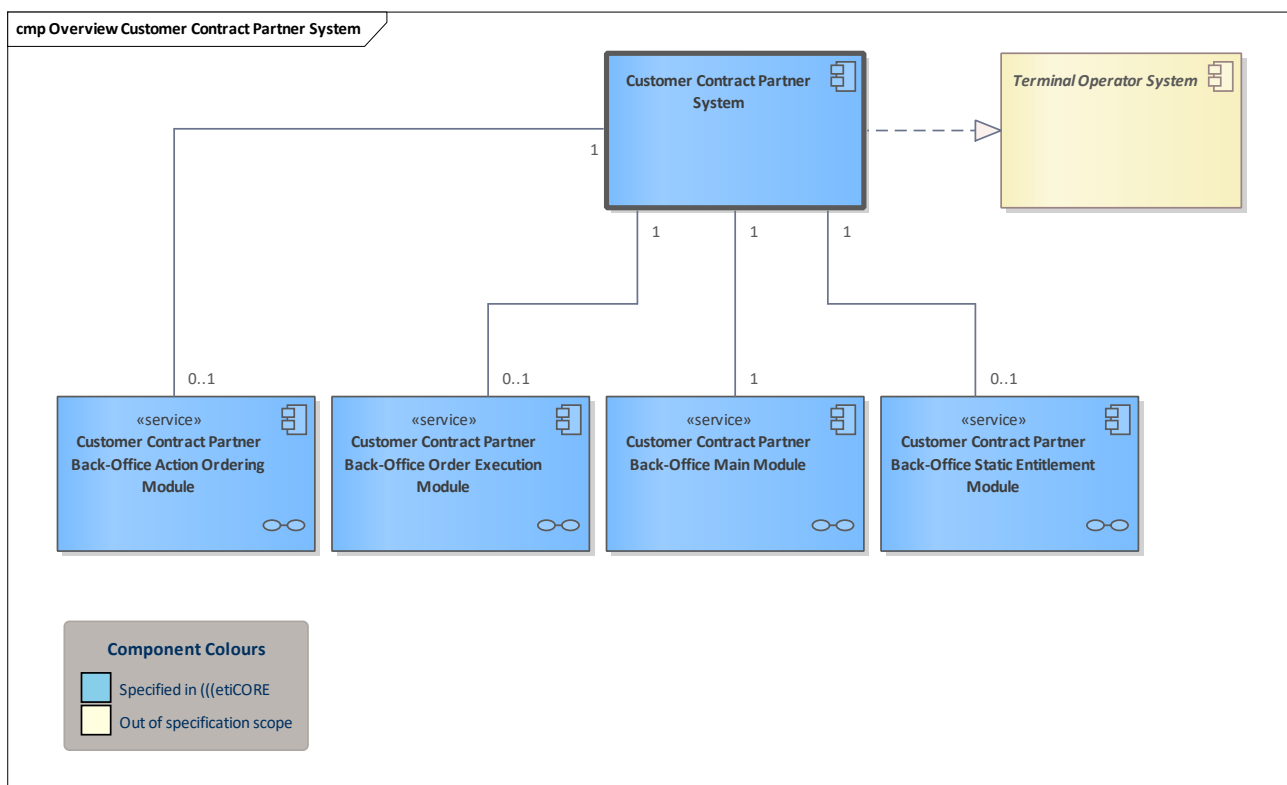


Figure 1: Overview Customer Contract Partner System

Shows the composition of a [Customer Contract Partner System](#).

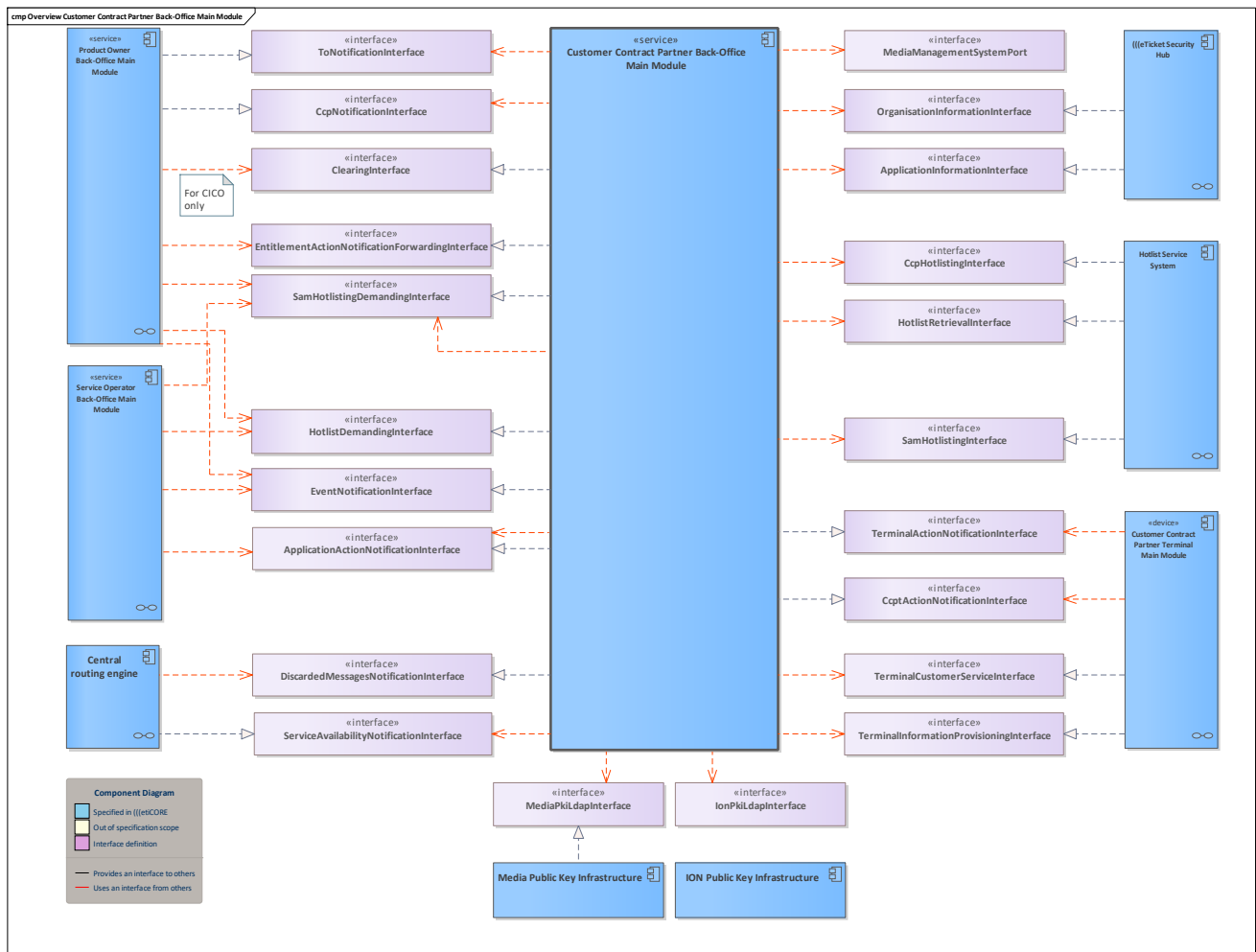


Figure 2: Overview Customer Contract Partner Back-Office Main Module  
Shows the interaction of a [Customer Contract Partner Back Office Main Module](#) via interfaces with other components.

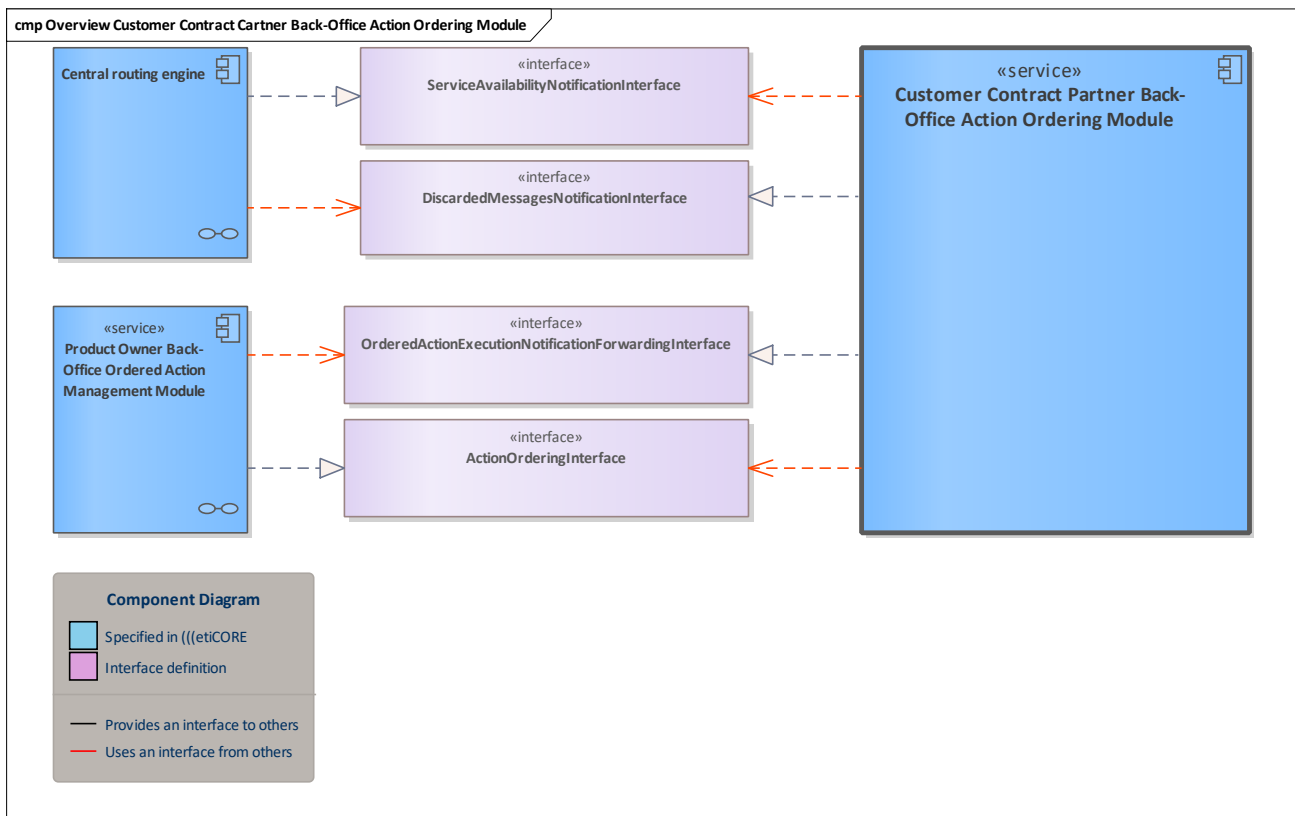


Figure 3: Overview Customer Contract Cartner Back-Office Action Ordering Module  
Shows the interaction of a [Customer Contract Partner Back Office Action Ordering Module](#) via interfaces with other components.

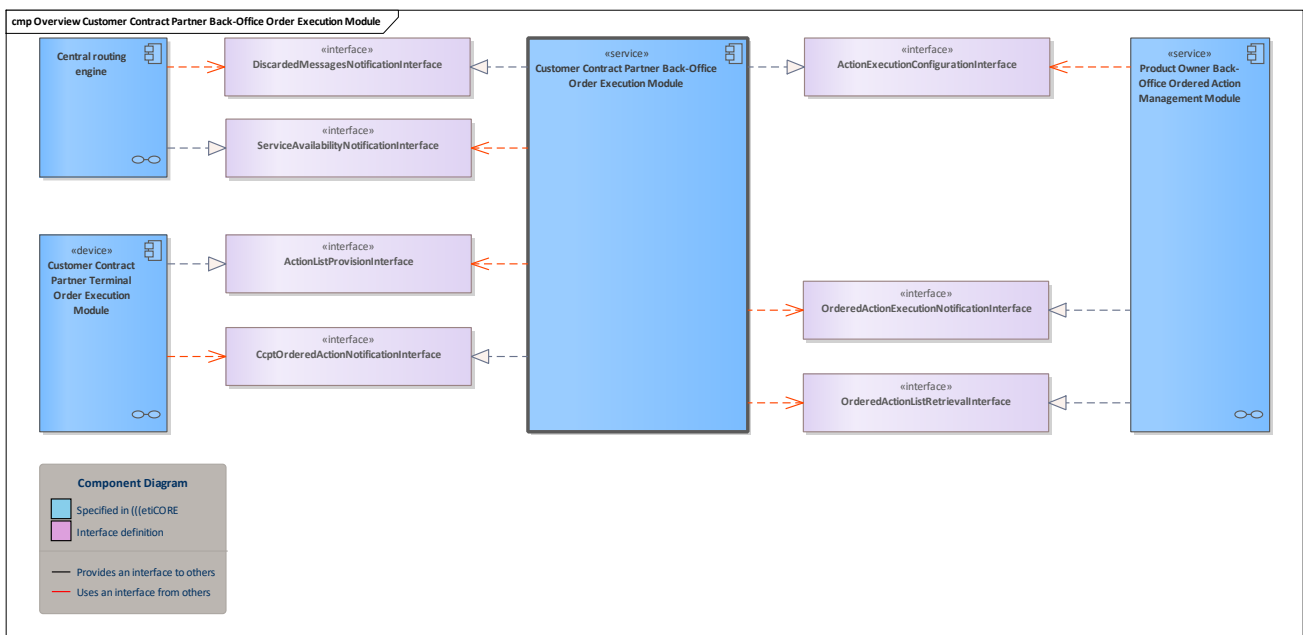


Figure 4: Overview Customer Contract Partner Back-Office Order Execution Module  
Shows the interaction of a [Customer Contract Partner Back Office Order Execution Module](#) via interfaces with other components.

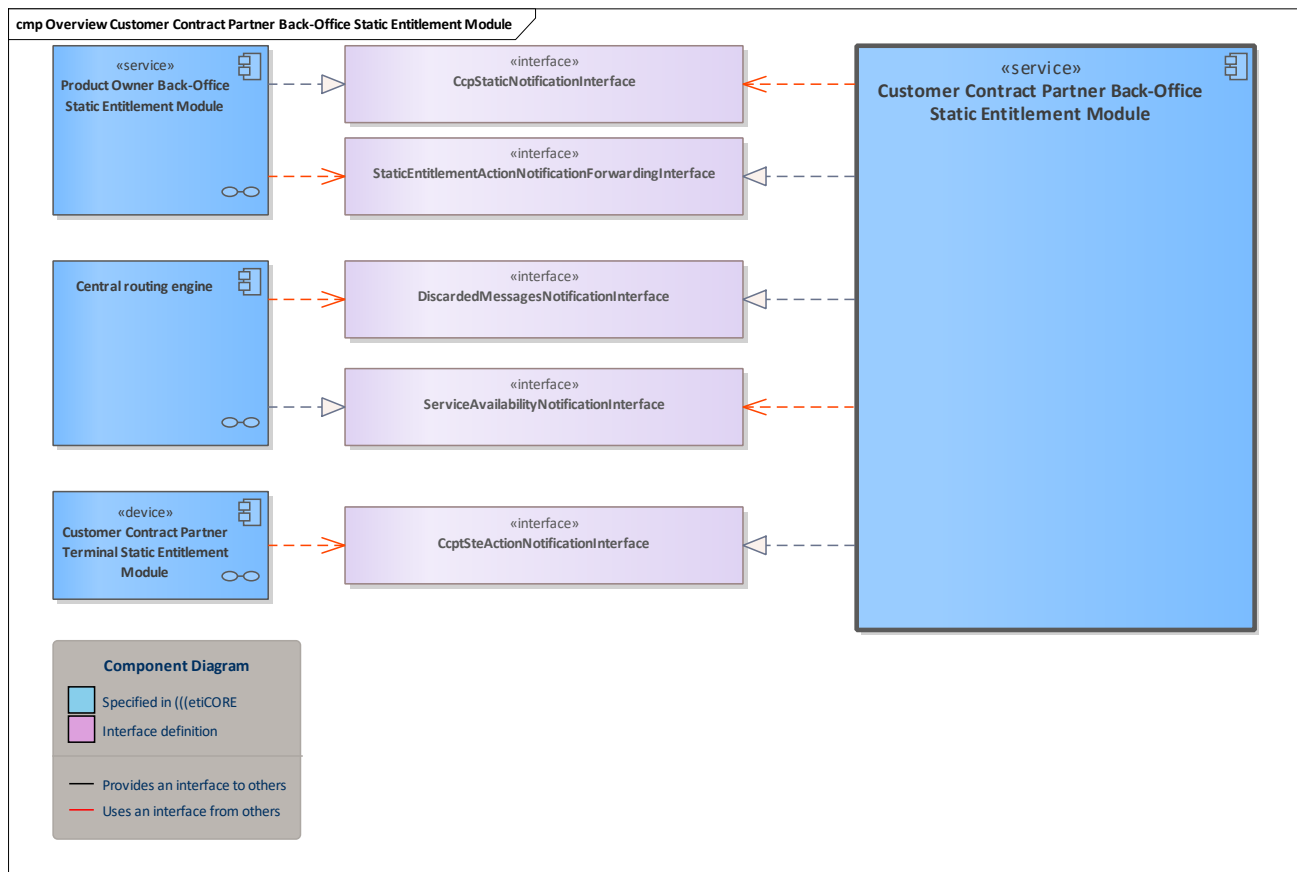


Figure 5: Overview Customer Contract Partner Back-Office Static Entitlement Module

Shows the interaction of a [Customer Contract Partner Back Office Static Entitlement Module](#) via interfaces with other components.

## 2.1 Customer Contract Partner System

Component for a customer contract partner system. Depending on the implemented functionality bundles, the customer contract partner system consists of

- [Customer Contract Partner Back Office Main Module](#) (always)
- [Customer Contract Partner Back Office Action Ordering Module](#) (if the CCP can order actions to be executed on user media later on terminals via action lists)
- [Customer Contract Partner Back Office Order Execution Module](#) (if the CCP is responsible for executing actions on user media via its terminals)
- [Customer Contract Partner Back Office Static Entitlement Module](#) (if the CCP also works with static entitlements)

## 2.2 Customer Contract Partner Back-Office Main Module

System (component) which implements the basic functionality required for the [Customer Contract Partner](#) and implements interfaces for

- another customer contract partner,
- the [Scheme Manager System](#),
- [Service Operator System](#) and
- [Product Owner System](#).



## 2.3 Customer Contract Partner Back-Office Action Ordering Module

System (component) which implements the functionality required for the [Customer Contract Partner](#) that orders actions on user media to be executed on terminals via action lists.

## 2.4 Customer Contract Partner Back-Office Order Execution Module

System (component) which implements the functionality required for the [Customer Contract Partner](#) that executes actions on user media on its terminals via action lists.

## 2.5 Customer Contract Partner Back-Office Static Entitlement Module

System (component) which implements the functionality required for the [Customer Contract Partner](#) and [Customer Contract Partner System](#) that works with static entitlements.

# 3 Notification Process Patterns

This chapter deals with the patterns which are widely used in the model, especially for notification processes.

If you understand these patterns, you will understand the structure of most of the use cases.

The different actions that can be executed by the user medium fall into different categories with respect to who is authorised to execute them. The important distinction here is whether the executing organisation is also the owner of the target object (UM application for actions targeting the application, entitlement for actions targeting the entitlement).

For some actions, the executing organisation is never the object owner (i.e. for all actions only relevant to Service Operators, which are never owners of UMs or entitlements). Others may only be performed by the owner, i.e. unblocking (outside of ordered action execution). The rest may be performed by organisations that may or may not be the object owner, e.g. debiting a payment method or blocking entitlement or application.

The actions (and their notifications) fall into four different categories :

- Entitlement owned
- Entitlement non-owned
- Application owned
- Application non-owned

Depending on the category the action falls into, the handling of their notifications with respect to potential forwarding to the owner and with respect to when the contractual processing happens may be different.

Templates for the execution of actions and the handling of their notifications for the categories entitlement owned and non-owned, as well as application owned and non-owned are shown.

Within these categories, the notification handling is very similar.

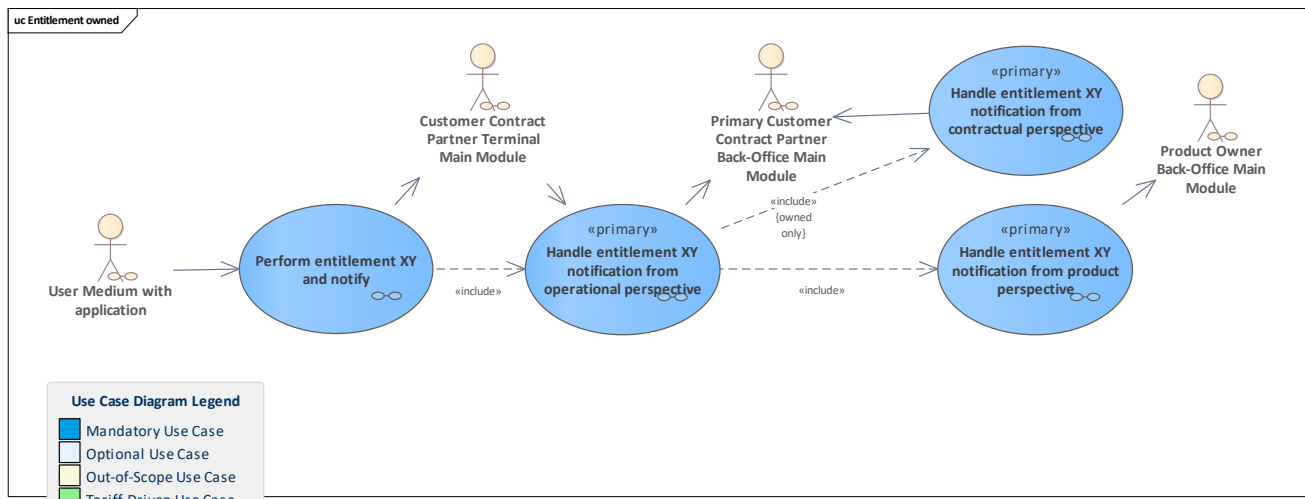


Figure 6: Entitlement owned

An entitlement-specific notification can only be created by the owner of the entitlement. The PO is informed about the action by the party performing the action, which always is the owner in this scenario.

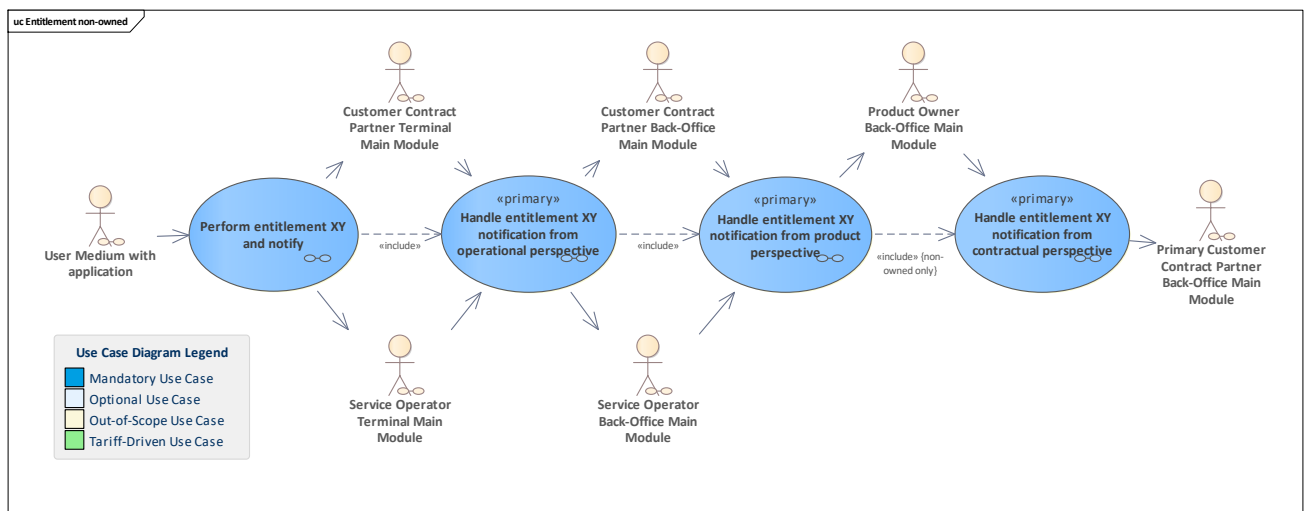


Figure 7: Entitlement non-owned

An entitlement-specific notification can only be created by parties other than the owner of the entitlement. The PO is informed about the action by the party performing the action. The owner is informed about the action by the PO.

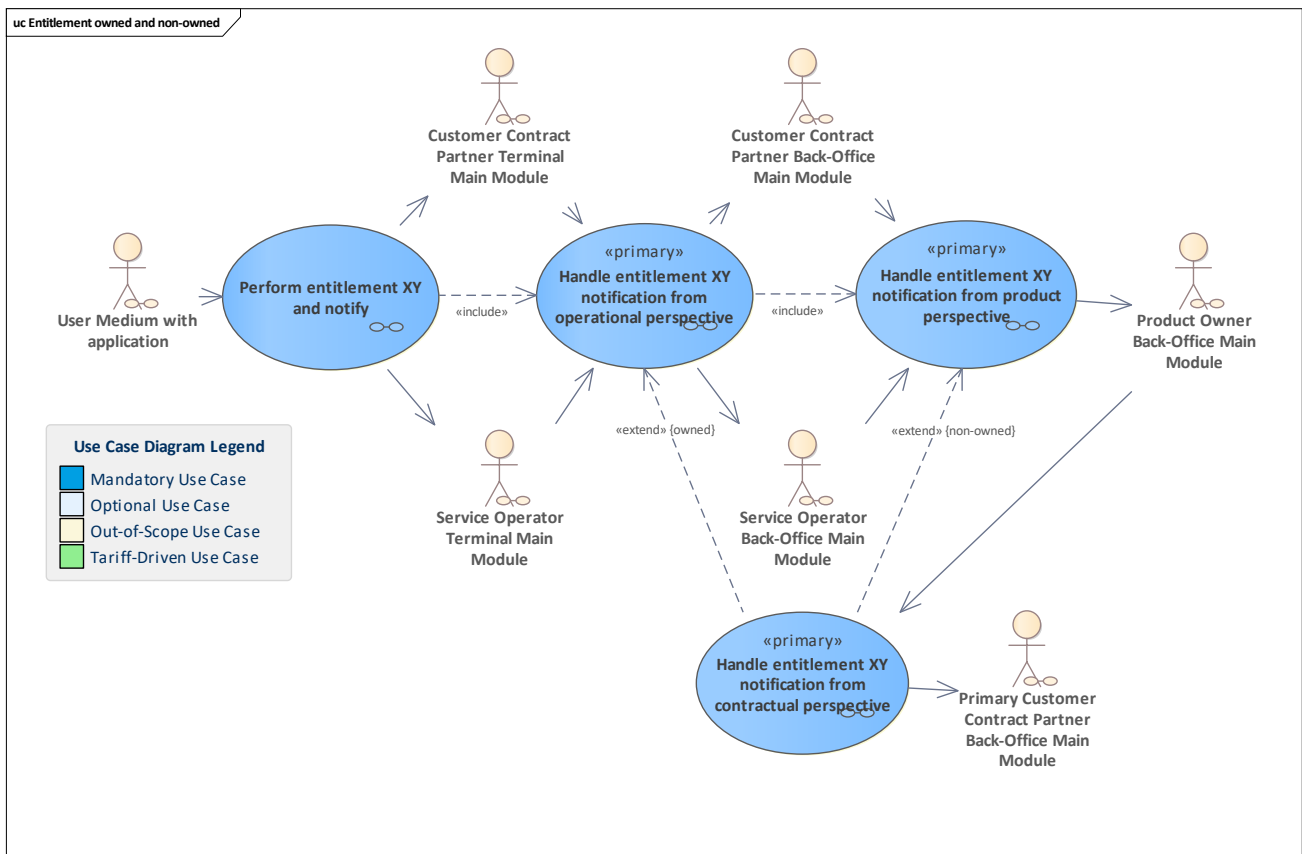


Figure 8: Entitlement owned and non-owned

An entitlement-specific notification can be created by the owner of the entitlement and other parties.

The PO is informed about the action by the party performing the action.

The owner is informed about the action by the PO if he did not perform the action himself.

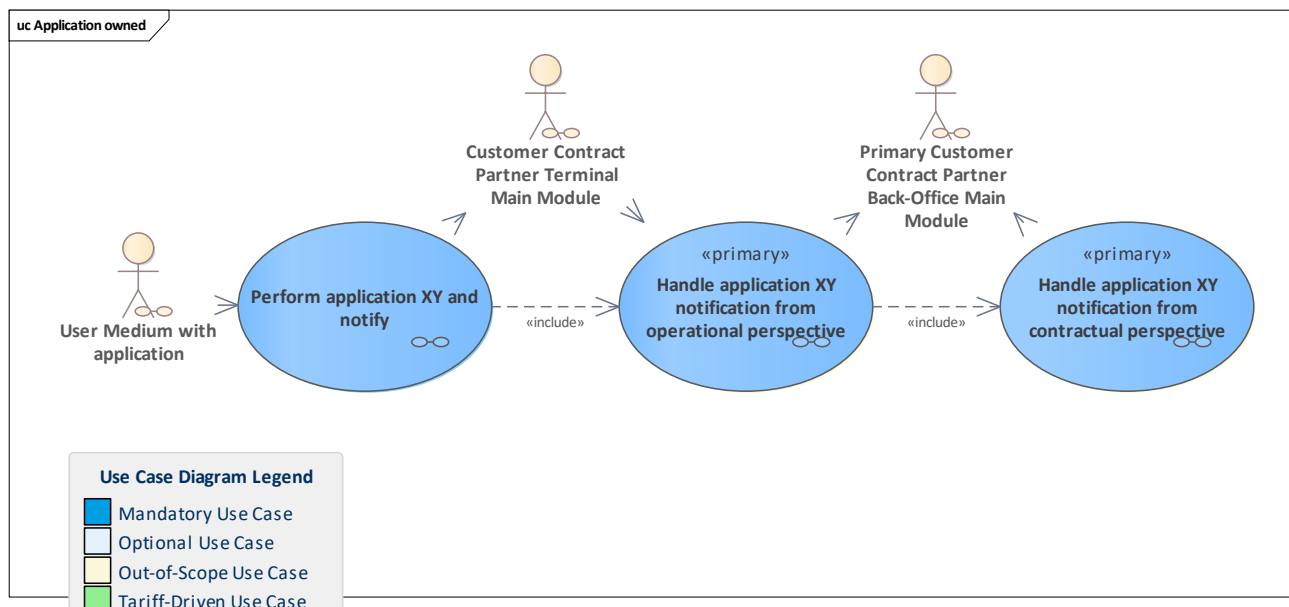


Figure 9: Application owned

An application-specific notification can only be created by the owner of the application.

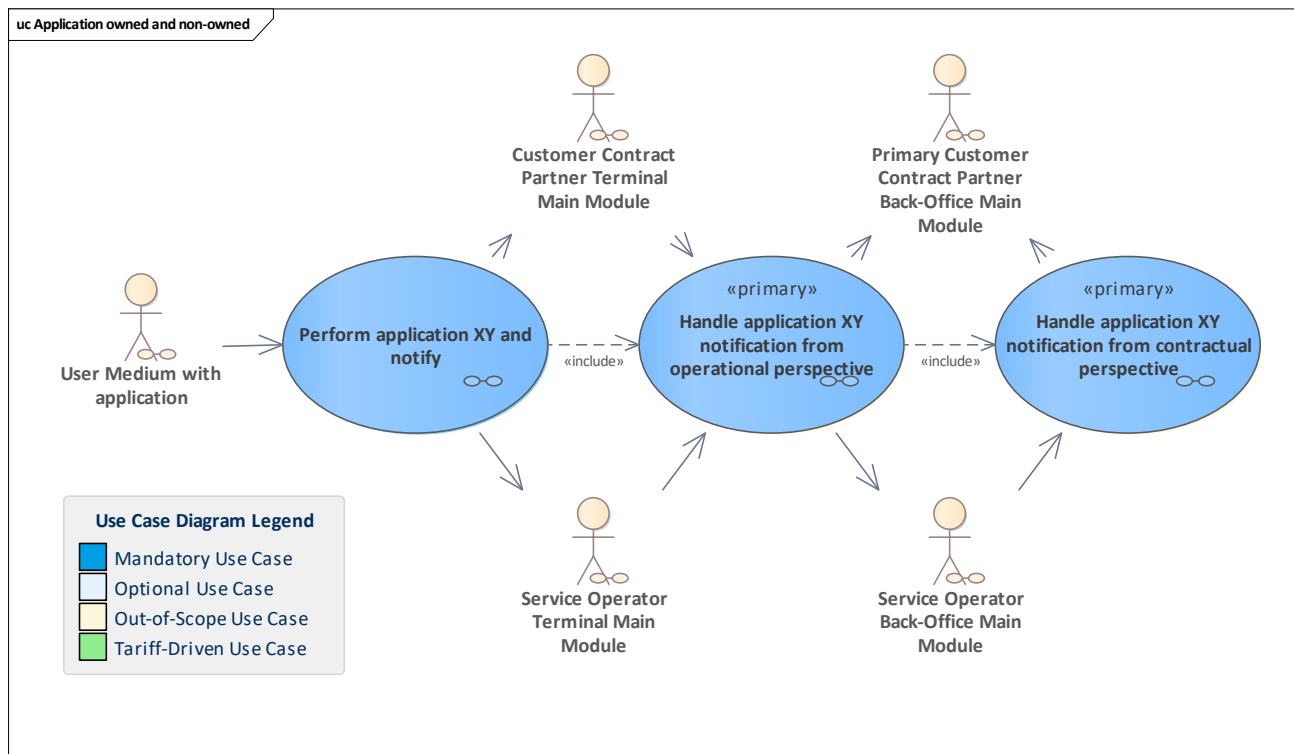


Figure 10: Application owned and non-owned

An application-specific notification can be created by the owner of the application and other parties.

The owner is informed about the action by the party performing the action if he did not perform the action himself.

## 3.1 Supporting activities

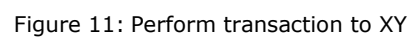
This chapter contains activities that are used in more than one use case.

### 3.1.1 Perform transaction to XY

Shows the actual transaction including XY. May start by ensuring, that a session is established. This is left out if there is a business requirement to securely retrieve the entitlement in question before performing the operation since that already had to be done in a session. The next step is to prepare the XY parameters for the action to be performed. Finally, it contains a transaction following the schema described in [Transaction](#).

In some special cases, a transaction may involve more than one action execution. In that case, there will be a variation of this diagram type leaving out the commit transaction step at the end (and the corresponding timeout error situation). This variation is then used in conjunction with a classic version of this diagram type within a single [Role-T-Module::Perform entitlement XY and notify](#) diagram.







### 3.1.1.2 SAM::Authorise XY

Call the SAM operation AUTHORISE\_ENTITLEMENT/AUTHORISE\_ACTION using the prepared parameters.

Since the SAM performs a roll-back in case of an error, no special error handling is shown here. If the SAM configuration does not allow the transaction to be completed, the error scenario "Action aborted" should be followed. This is not shown here, since the terminal should be able to anticipate this situation, e.g. by retrieving the SAM product issuance rights before trying to AUTHORISE\_ENTITLEMENT to make sure the SAM is configured to issue the product in question.

### 3.1.1.3 SAM::Validate XY and authorise commit

Call the SAM operation VALIDATE\_ACTION using the secured attestation prepared by the UM. The only relevant error code in this situation is the error code *SM data objects incorrect*. All other error codes should either not happen at all or be easy to correct by the terminal. If that is not true for any reason, proceed as if the action had to be aborted.

If the connection to the SAM or UM is lost during the execution of this command, the transaction is also aborted.

### 3.1.1.4 UM::Commit transaction

Call the UM operation COMMIT\_TRANSACTION using the secured attestation prepared by the UM.

The only relevant error code in this situation is the error code *SM data objects incorrect*. All other error codes should either not happen at all or be easy to correct by the terminal. If that is not true for any reason, proceed as if the action had to be aborted.

If the connection to the UM is lost during the execution of this command such that it cannot be ruled out that the command successfully ran on the UM side (and only the response was not transmitted), the *commit timeout* error scenario has to be followed.

### 3.1.1.5 UM::Execute XY

Call the UM operation ISSUE\_ENTITLEMENT / EXECUTE\_ACTION using the Command APDU readily prepared by the SAM.

In case of an error, the terminal cannot correct the message to eliminate the problem since it is, and has to be, secured (encrypted and MAC-signed) using the session keys only SAM and UM know. Thus, the operation has to be re-authorised requiring the use of further SAM counters, effectively aborting the transaction since it may - on a business level - also affect other actions within the same transaction or may already have (in case of the error code *SM data objects incorrect*) reset the session on the UM side. For the sake of simplicity, this is not distinguished here and the whole transaction is aborted.

If the connection to the SAM or UM is lost during the execution of this command, the transaction is also aborted.

### 3.1.1.6 Update shadow copies of SAM counters

Update the shadow copies of the SAM counters.

For AUTHORISE\_ACTION this means incrementing the SAM action counter by one.

For AUTHORISE\_ENTITLEMENT this means incrementing the SAM entitlement issuance counter and the product issuance counter corresponding to the issued product by one.

### 3.1.2 Prepare XY parameters

Composes the parameters of the UM operation (ISSUE\_ENTITLEMENT/EXECUTE\_ACTION) that are set by the terminal for action parameters. These parameters are then given to the authorising operation of the SAM (AUTHORISE\_ENTITLEMENT/AUTHORISE\_ACTION) and further adjusted by the SAM yielding a fully prepared Command APDU to be used for the UM.

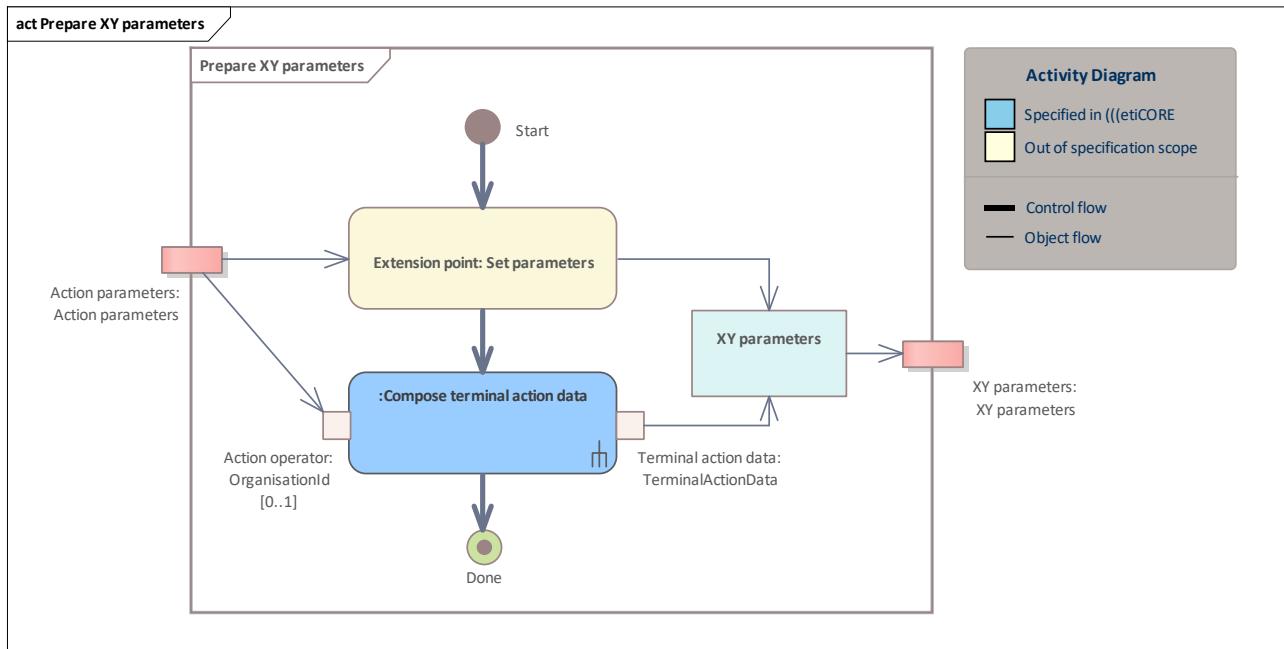


Figure 12: Prepare XY parameters

#### 3.1.2.1 Extension point: Set parameters

This extension point is a placeholder for the use case-specific preparatory steps to create an action parameters.

For entitlement actions, the product ID given via the parameters is always part of the action parameters.

For application actions, a pseudo product ID is constructed using the UM Owner ID as the PO Org ID.

## 3.2 Supporting classes

Gives an overview of the supporting classes used for the pattern diagrams.

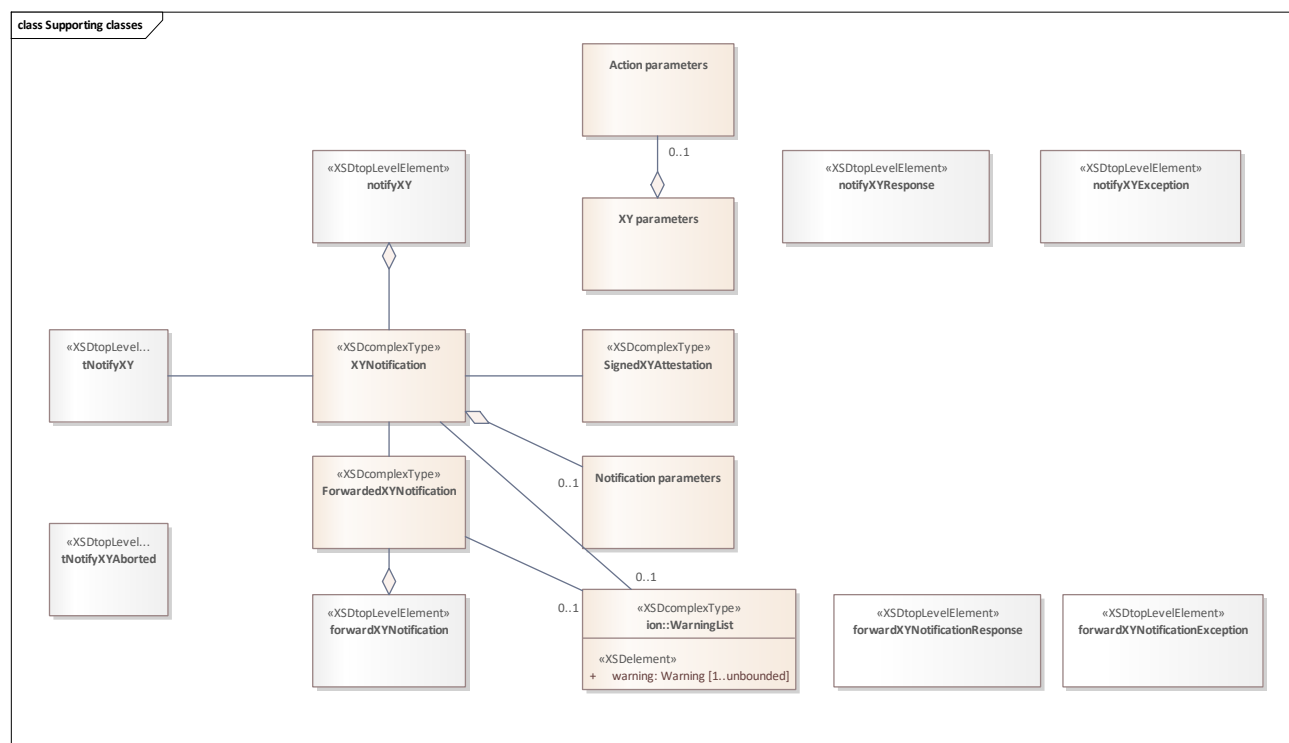


Figure 13: Supporting classes

### 3.2.1 Action parameters

Only applicable to entitlement actions.

Always contains the entry ID and the product ID.

May contain additional, use case-specific parameters, e.g. action tariff parameters.

### 3.2.2 XY parameters

Parameters for the action used to call the SAM operation authorising the action on the UM side.

### 3.2.3 SignedXYAttestation

An attestation of successful action execution signed by the UM involved.

Also identifies the UM so that back-office systems can retrieve the corresponding certificate and verify the signature.

### 3.2.4 Notification parameters

Additional parameters to insert into the notification bearing the signed attestation.



### 3.2.5 XYNotification

Every XY notification consists of the following three building blocks:

- an XY attestation signed by the UM
- for some use cases: additional information (see [Notification parameters](#))
- optional: a warning list

### 3.2.6 ForwardedXYNotification

A forwarded XY notification is a wrapper around the XY notification that may carry additional events in a separate event list.

### 3.2.7 tNotifyXY

Element used to transmit an XY notification from the terminal to its back-office system.

### 3.2.8 tNotifyXYAborted

Element used to transmit the information about an aborted XY action from the terminal to its back-office system.

### 3.2.9 notifyXY

Element used to inform the product owner about an entitlement action execution and to inform the owner of an application about an application action execution.

### 3.2.10 notifyXYResponse

Element used in response to [notifyXY](#) for normal process termination.

### 3.2.11 notifyXYException

Element used in reply to [notifyXY](#) for abnormal process termination.

### 3.2.12 forwardXYNotification

Element used to inform the entitlement owner about an entitlement action execution by a third party.

### 3.2.13 forwardXYNotificationResponse

Element used in response to [forwardXYNotification](#) for normal process termination.

### 3.2.14 forwardXYNotificationException

Element used in reply to [forwardXYNotification](#) for abnormal process termination.

## 3.3 Perform entitlement XY and notify

See [Perform entitlement XY and notify](#)

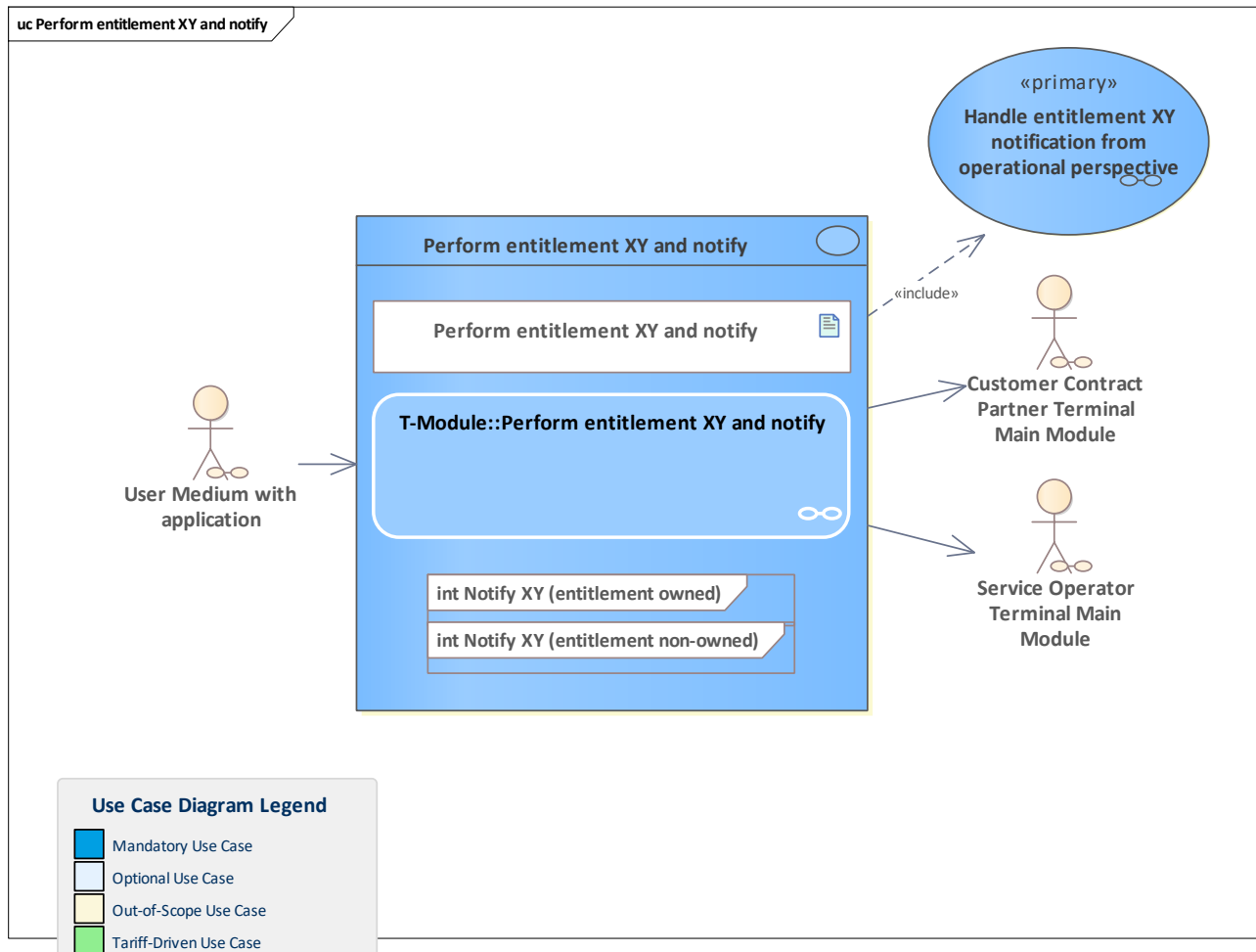


Figure 14: Perform entitlement XY and notify

### 3.3.1 Perform entitlement XY and notify

The user medium-based entitlement action XY is performed by the terminal in conjunction with a SAM and the back-office system corresponding to the terminal is notified about the action execution.

Combines the transaction including the action XY and the notification processes triggered by this. Shows the interaction between the UM transaction and the notification processes. The timeout warning is generated in the corresponding diagram if needed.

The business preconditions to perform the transaction (e.g. determining the actual need to block an entitlement or making sure that a stored-value payment method is sufficiently charged) are already satisfied before the process shown in this diagram begins.

**Note:**

Executing orders in the context of ordered action management involves the same UM operations as outside of that scope, but leads to different notification processes (partly belonging to different notification categories). In this case, only one diagram "Perform transaction to XY" may be needed, which is used in two diagrams "T-XYZ::Perform XY and notify" combining it with different notification activities (i.e. the ordered and the regular variant).

### 3.3.2 T-Module::Perform entitlement XY and notify

See [Perform entitlement XY and notify](#)

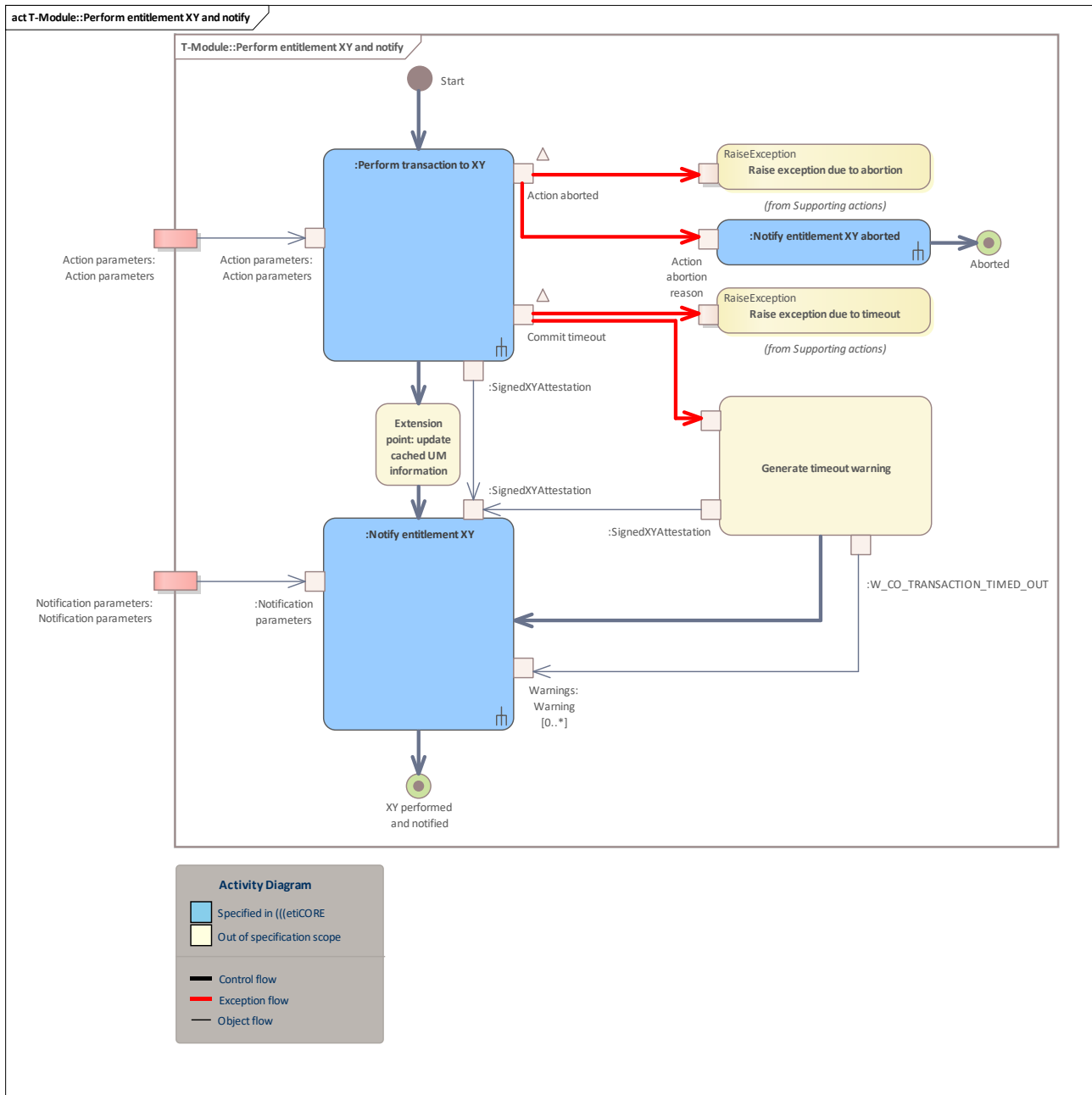


Figure 15: T-Module::Perform entitlement XY and notify

#### 3.3.2.1

See [Perform transaction to XY](#)

#### 3.3.2.2

See [Notify entitlement XY](#)

### 3.3.2.3

See [Notify entitlement XY aborted](#)

### 3.3.2.4 Extension point: update cached UM information

Update the cached information about the UM as appropriate for the executed action, e.g. entitlement action counter, application status, etc.

### 3.3.2.5 Generate timeout warning

Generate a timeout warning to be placed into the notification sent about the executed action.

## 3.3.3 Notify entitlement XY

The terminal notifies its back-office system about a successful action execution.

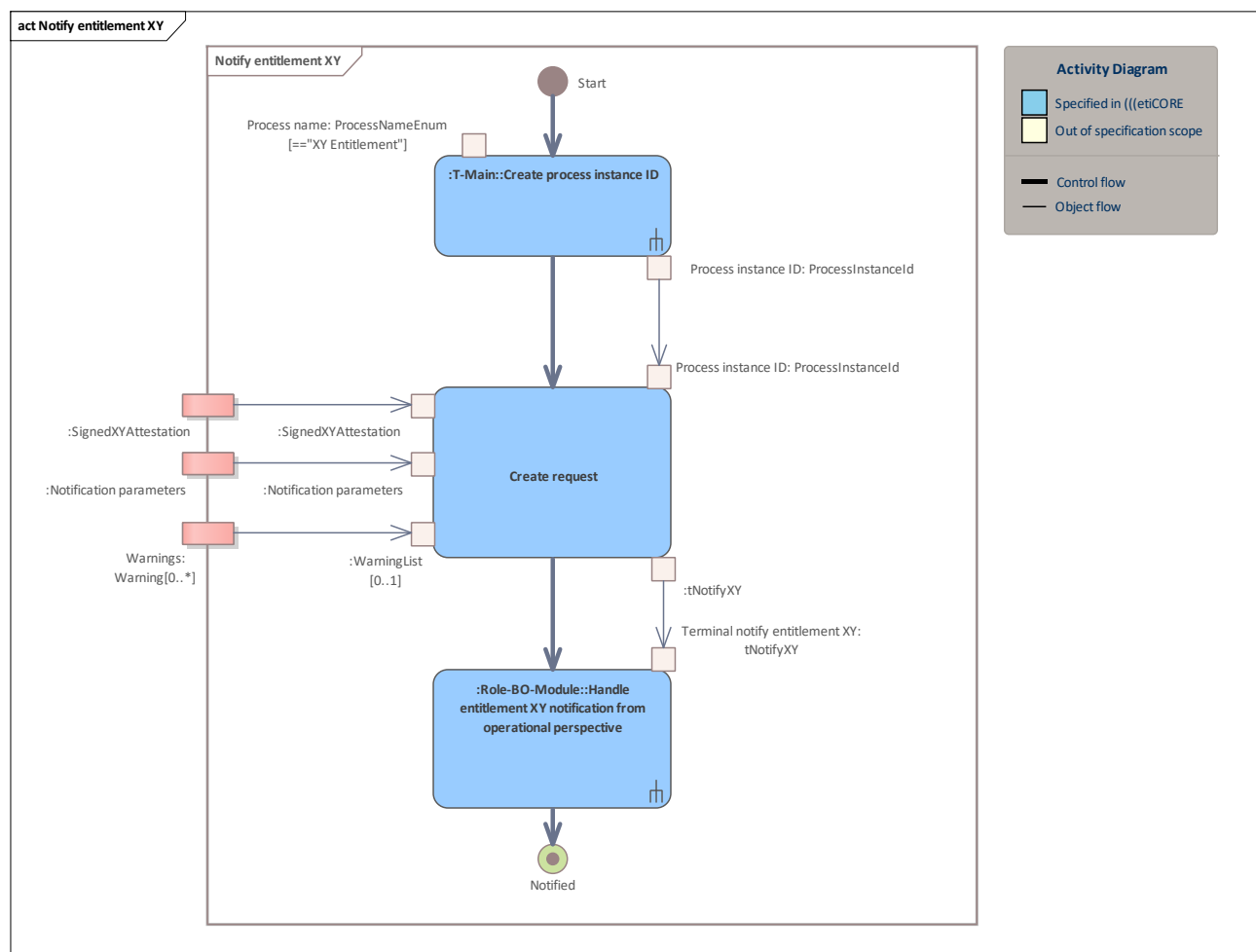


Figure 16: Notify entitlement XY



### 3.3.4 Notify entitlement XY aborted

The terminal notifies its back-office system about an aborted action.

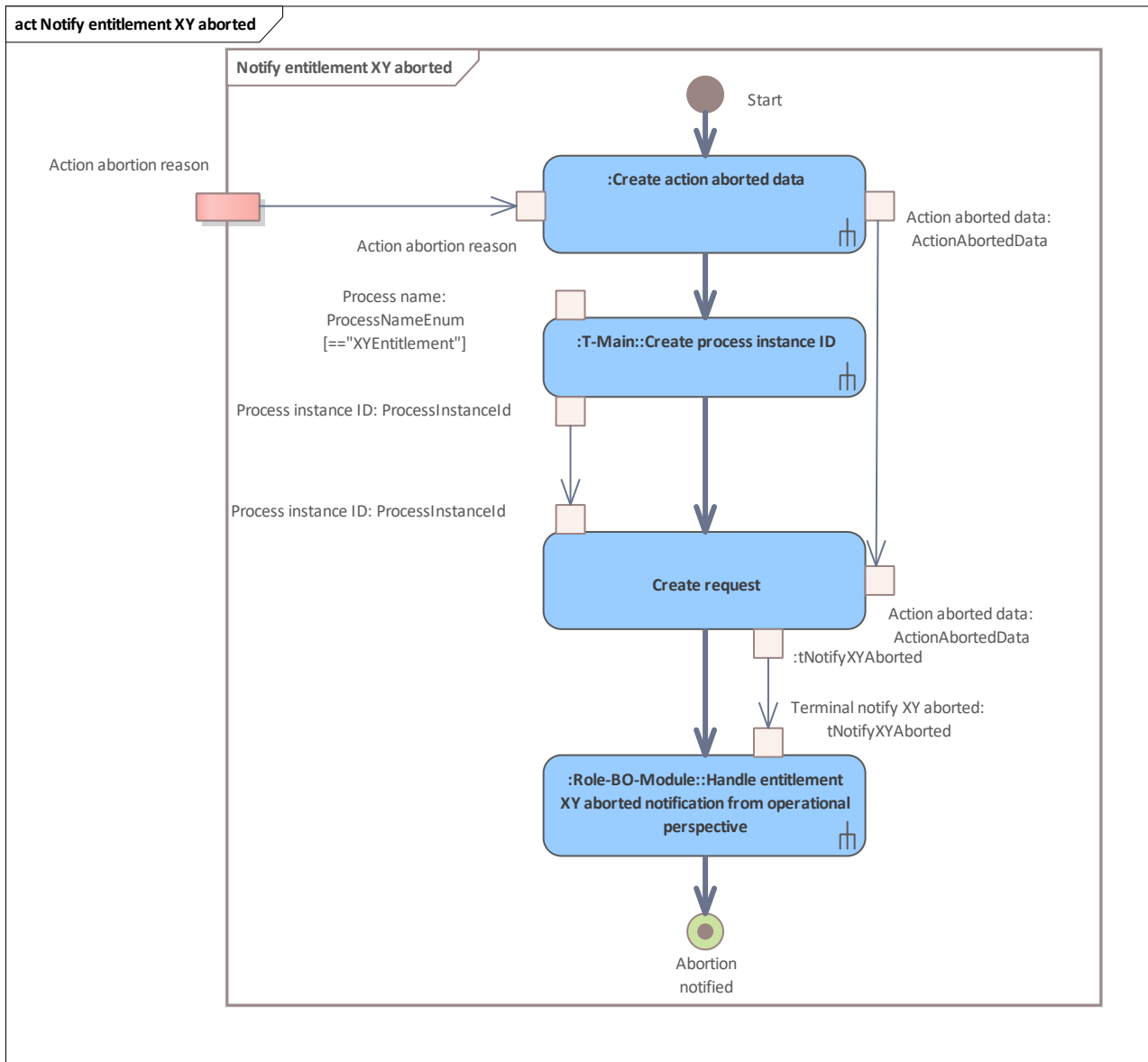


Figure 17: Notify entitlement XY aborted

### 3.3.5 Notify entitlement XY aborted based on attestation

If there are several actions within a transaction, this variant is to be used for the actions already completed (but not committed).

Since the attestation is already available, we can extract the required data from it instead of reproducing it.

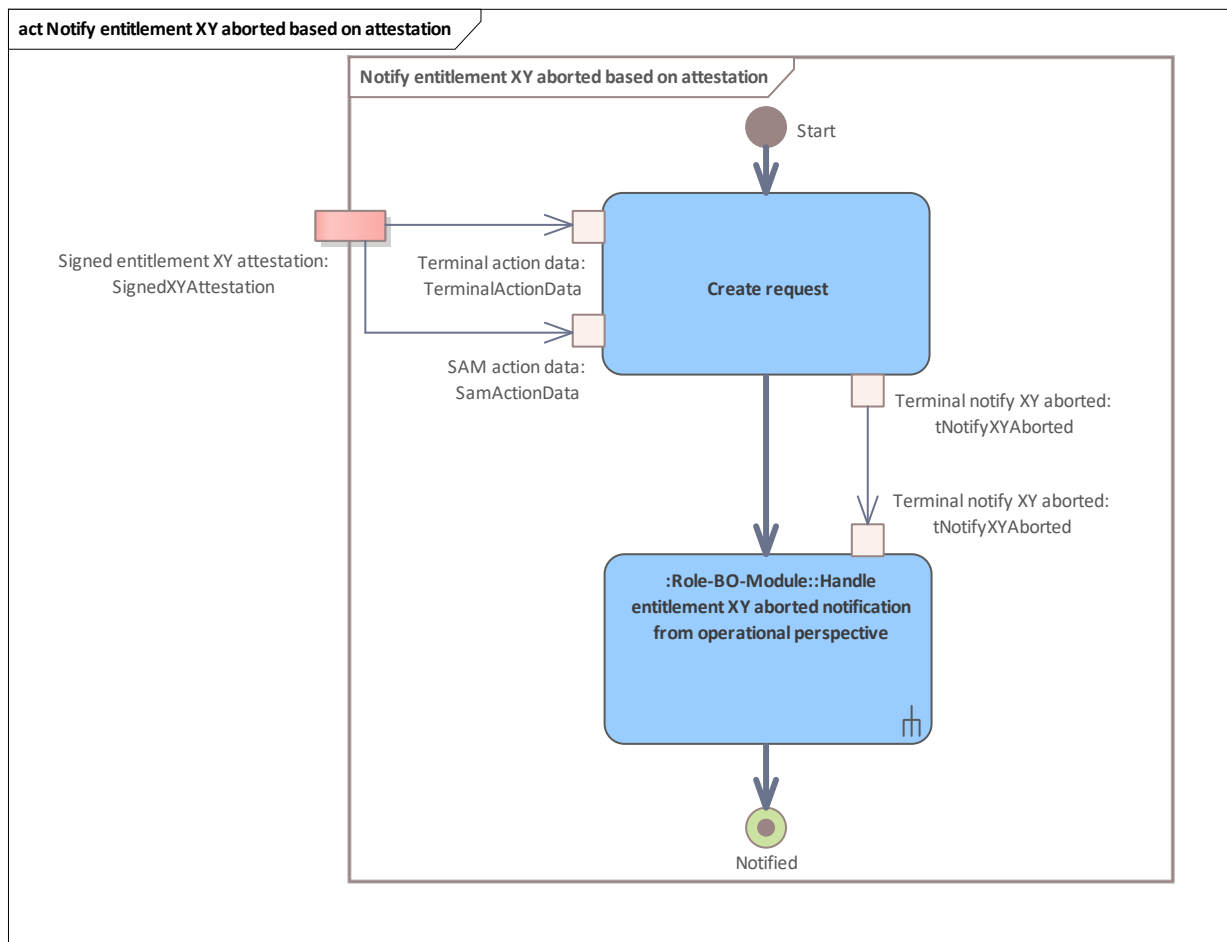


Figure 18: Notify entitlement XY aborted based on attestation

### 3.3.6 Notify XY (entitlement owned)

Shows the message chain starting in a terminal (normally customer contract partner terminal), sent to a terminal operator back-office system (primary customer contract partner system) and finally sent to the product owner system.

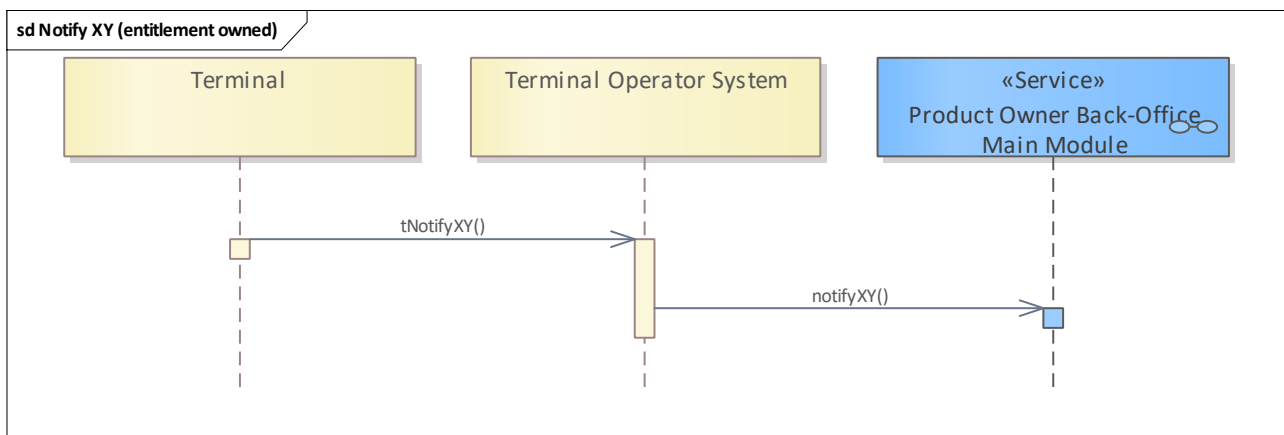


Figure 19: Notify XY (entitlement owned)

See [Notify XY \(entitlement owned\)](#)

### 3.3.7 Notify XY (entitlement non-owned)

Shows the message chain starting in a terminal (customer contract partner terminal or service operator terminal), sent to a terminal operator back-office system (customer contract partner system or service operator system) then sent to the product owner system and finally to the owning primary customer contract partner system.

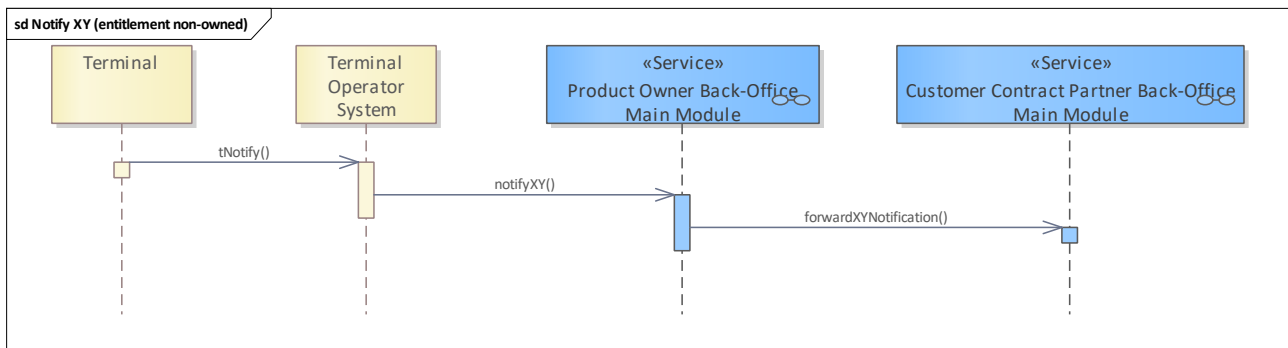


Figure 20: Notify XY (entitlement non-owned)

See [Notify XY \(entitlement non-owned\)](#).

## 3.4 Perform application XY and notify

See [Perform application XY and notify](#)

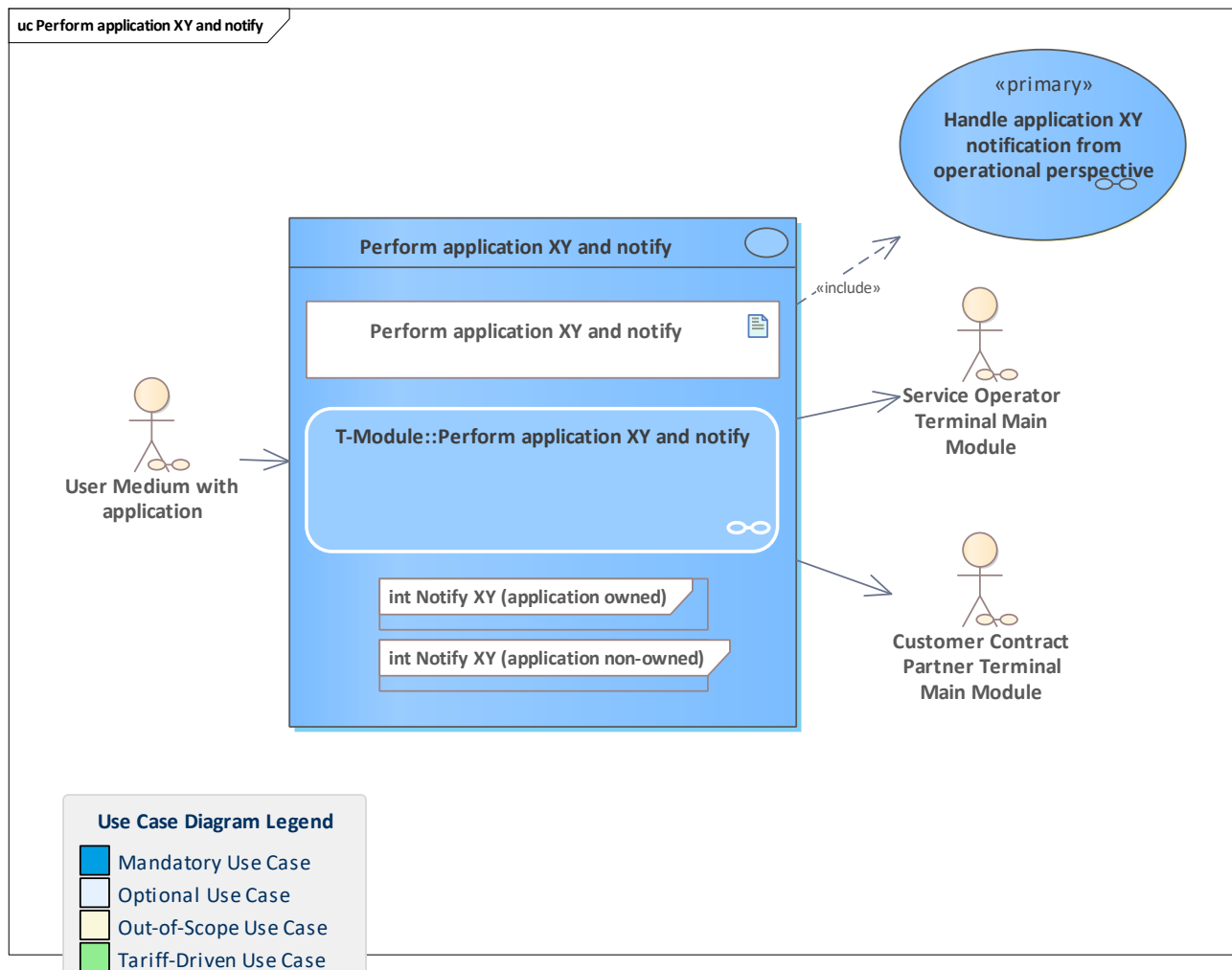


Figure 21: Perform application XY and notify

### 3.4.1 Perform application XY and notify

The UM application action XY is performed by the terminal in conjunction with a SAM and the back-office system corresponding to the terminal is notified about the action execution.

Combines the transaction including the action XY and the notification processes triggered by this. Shows the interaction between the UM transaction and the notification processes. The timeout warning is generated in this diagram if needed.

The business preconditions to perform the transaction (e.g. determining the actual need to block an application) are already satisfied before the process shown in this diagram begins.

### 3.4.2 T-Module::Perform application XY and notify

See [Perform application XY and notify](#)

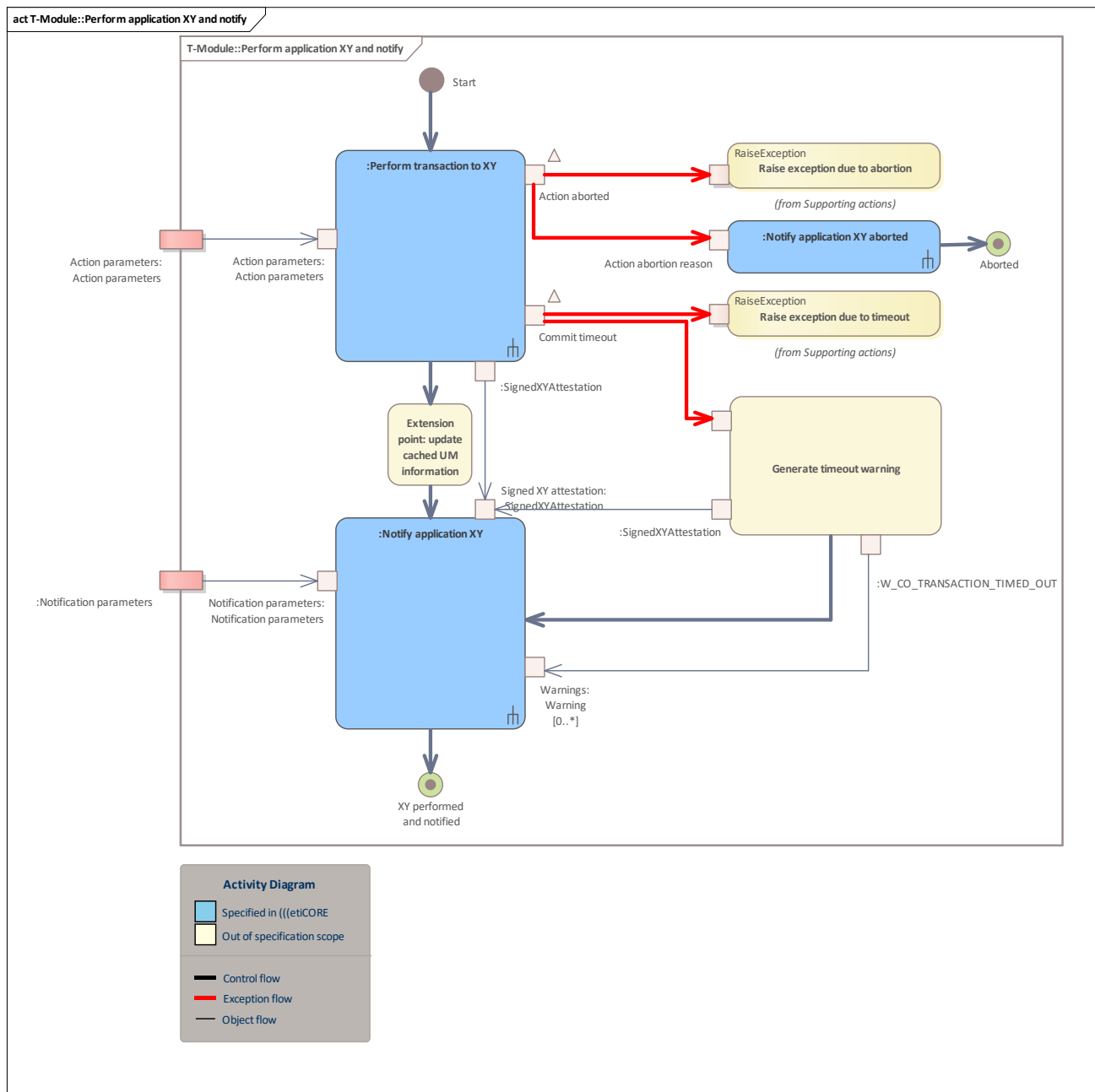


Figure 22: T-Module::Perform application XY and notify

### 3.4.2.1

See [Perform transaction to XY](#)

### 3.4.2.2

See [Notify entitlement XY aborted](#)

### 3.4.2.3

See [Notify entitlement XY](#)

### 3.4.2.4 Extension point: update cached UM information

Update the cached information about the UM as appropriate for the executed action, e.g. entitlement action counter, application status, etc.

### 3.4.2.5 Generate timeout warning

Generate a timeout warning to be placed into the notification sent about the executed action.

## 3.4.3 Notify application XY

The terminal notifies its back-office system about a successful action execution.

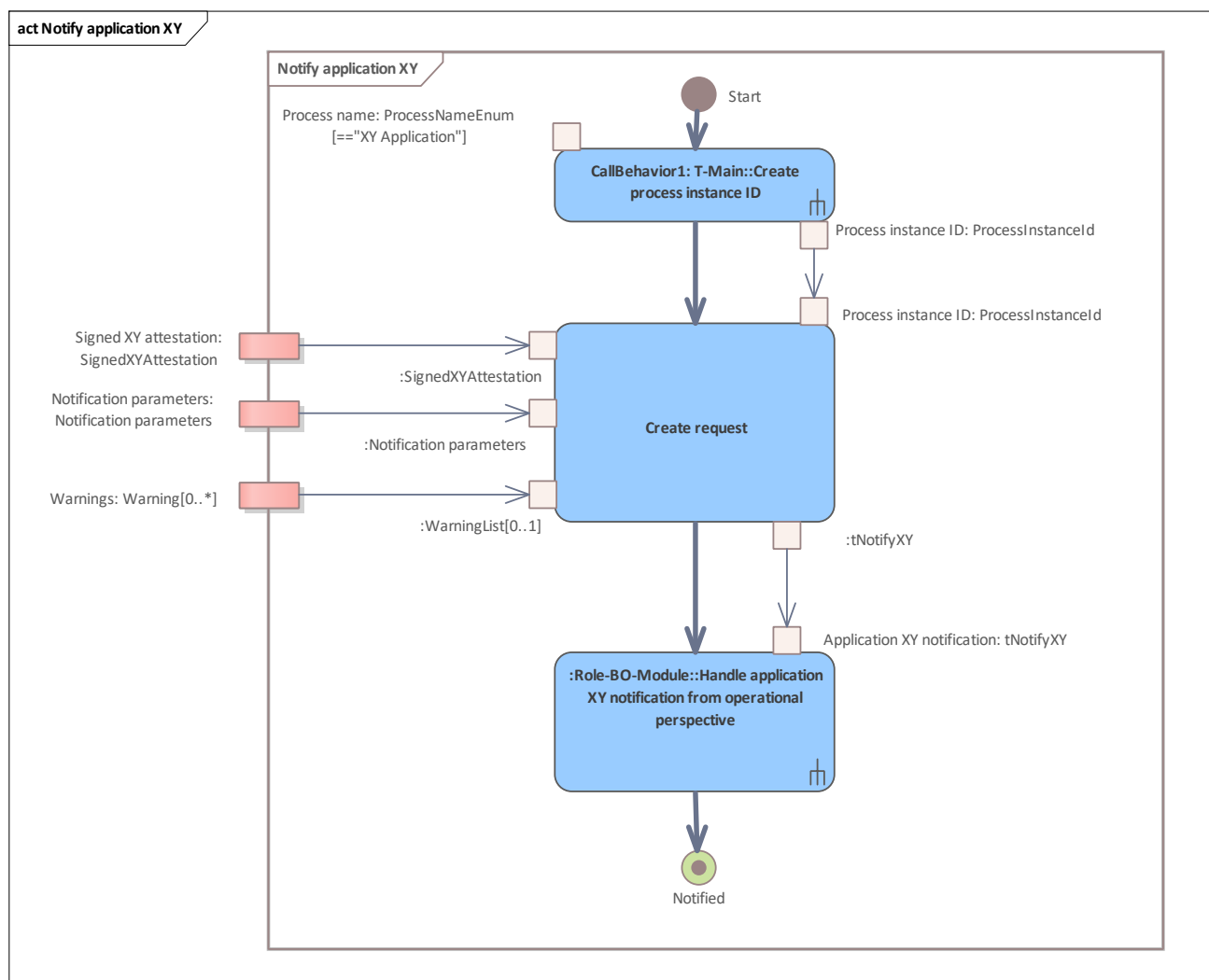


Figure 23: Notify application XY

### 3.4.4 Notify application XY aborted

The terminal notifies its back-office system about an aborted action.

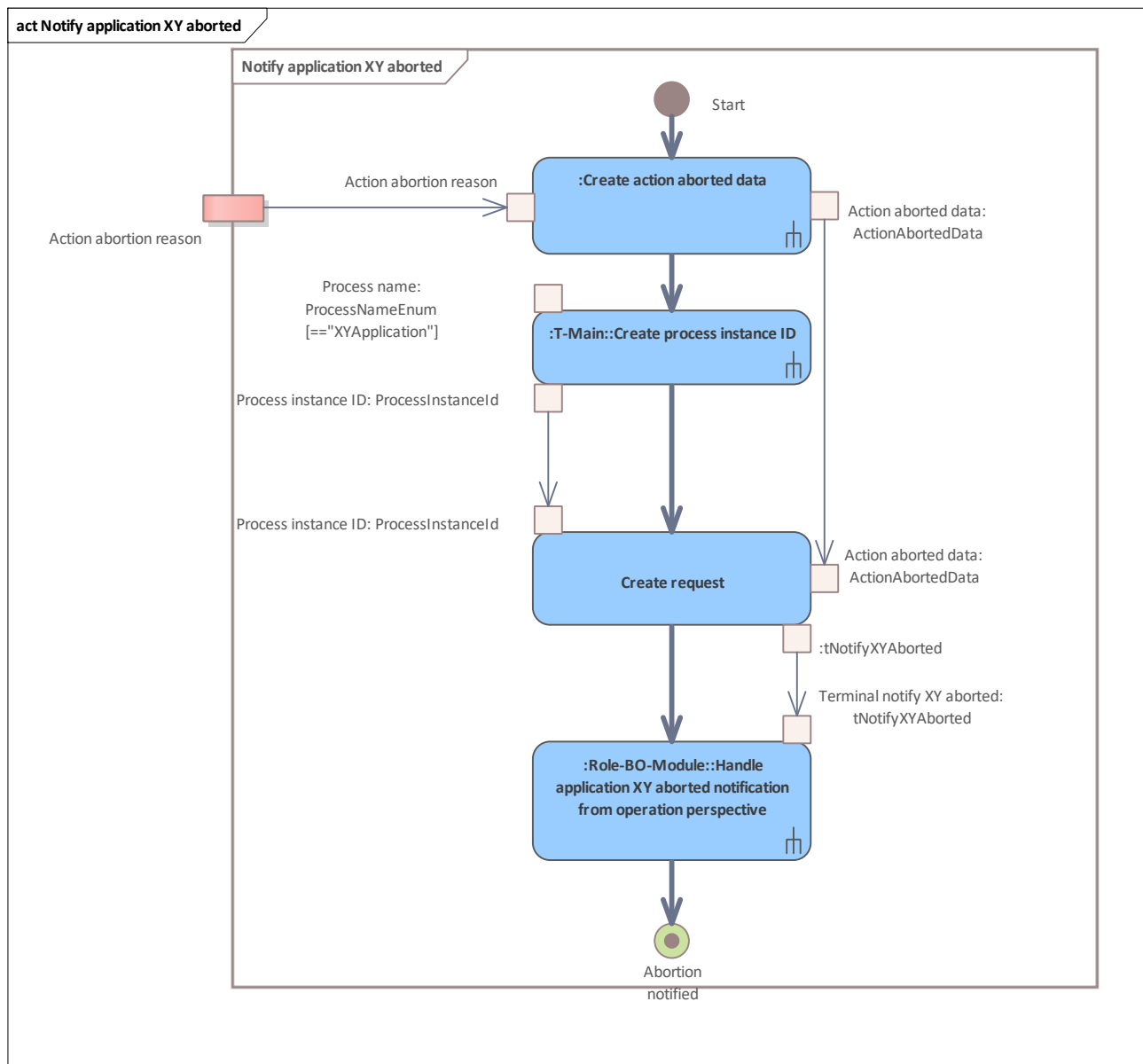


Figure 24: Notify application XY aborted

### 3.4.5 Notify XY (application owned)

Shows the message chain starting in a terminal (normally customer contract partner terminal), sent to a terminal operator back-office system (primary customer contract partner system).

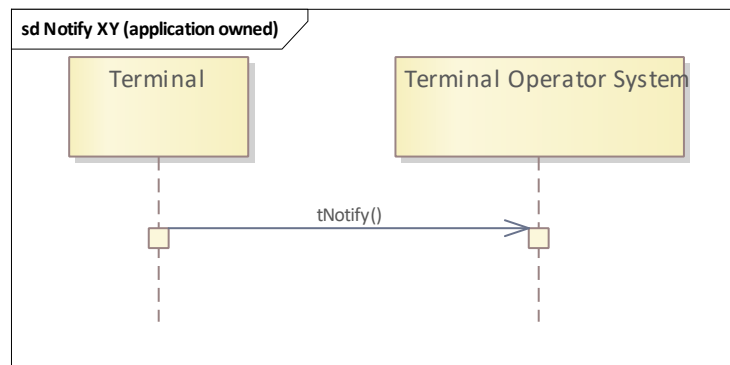


Figure 25: Notify XY (application owned)

See [Notify XY \(application owned\)](#).

### 3.4.6 Notify XY (application non-owned)

Shows the message chain starting in a terminal (customer contract partner terminal or service operator terminal), sent to a terminal operator back-office system (customer contract partner system or service operator system) and finally to the owning primary customer contract partner system.

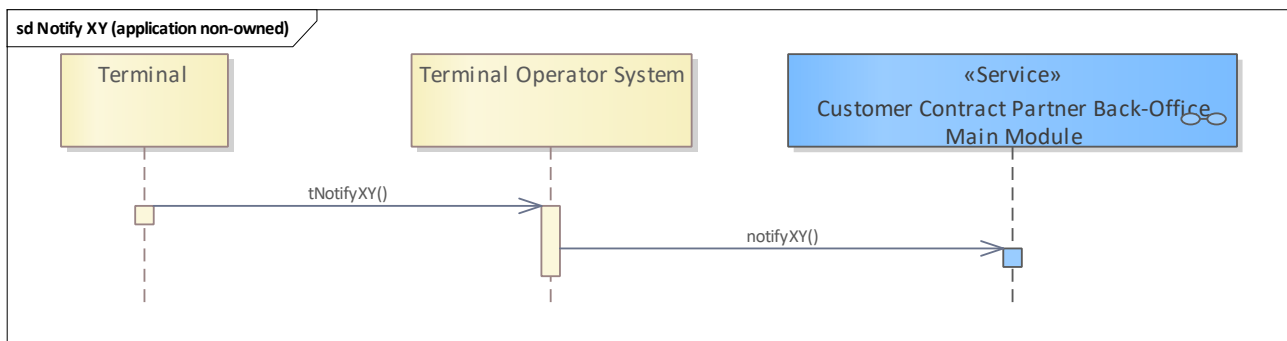


Figure 26: Notify XY (application non-owned)

See [Notify XY \(application non-owned\)](#).

## 3.5 Handle entitlement XY notification from operational perspective

See [Handle entitlement XY notification from operational perspective](#)



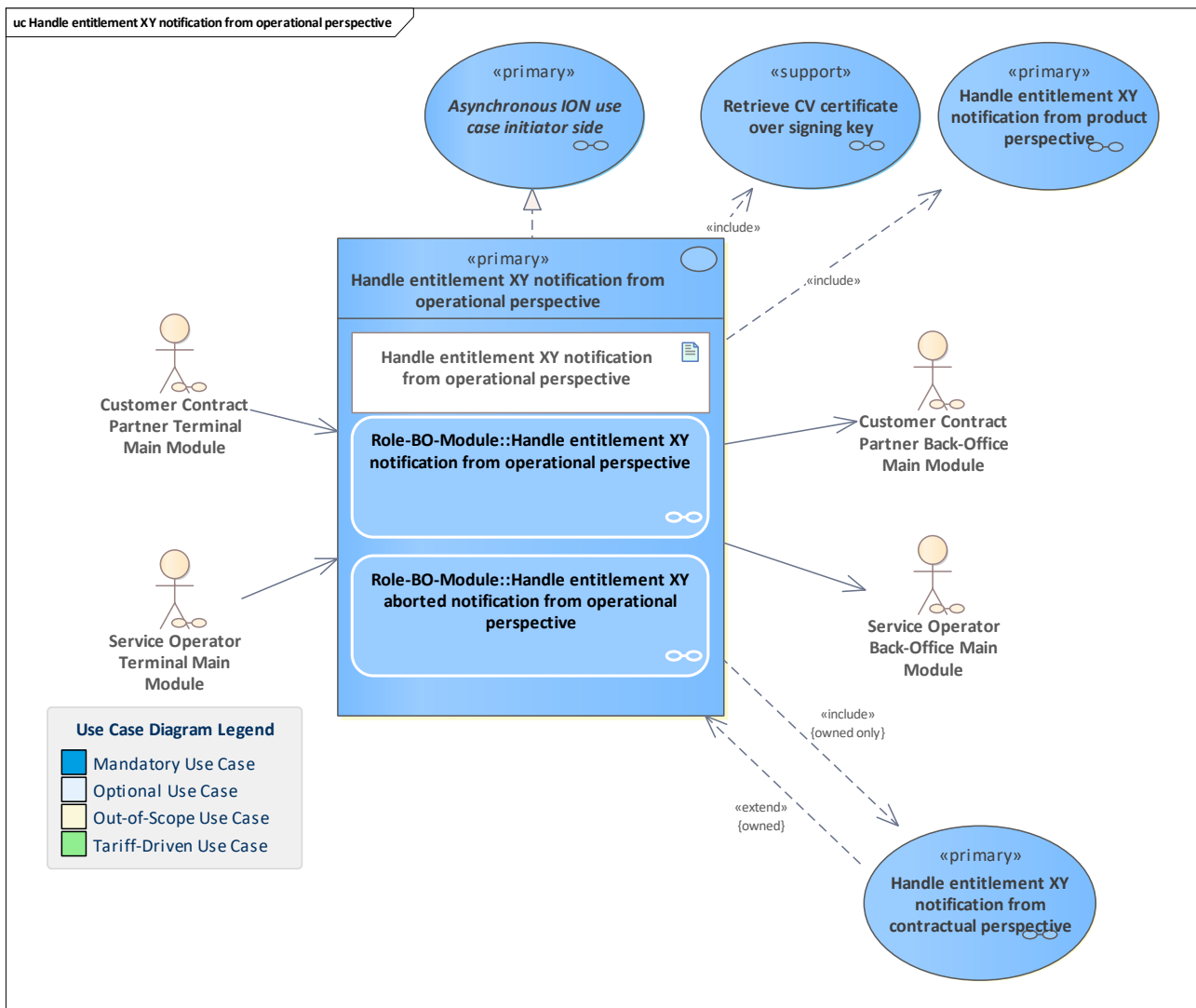


Figure 27: Handle entitlement XY notification from operational perspective

### 3.5.1 Handle entitlement XY notification from operational perspective

A back-office system belonging to a terminal that executed a UM action with an entitlement processes the notification about that action execution.

The processing is done from the operational perspective, focusing on the terminal-side of the action execution, i.e. logging the used SAM counter values.

Depending on the use case and the ownership of the entitlement the action was executed on, other back-office systems are informed about the action execution.

### 3.5.2 Role-BO-Module::Handle entitlement XY notification from operational perspective

See [Handle entitlement XY notification from operational perspective](#)

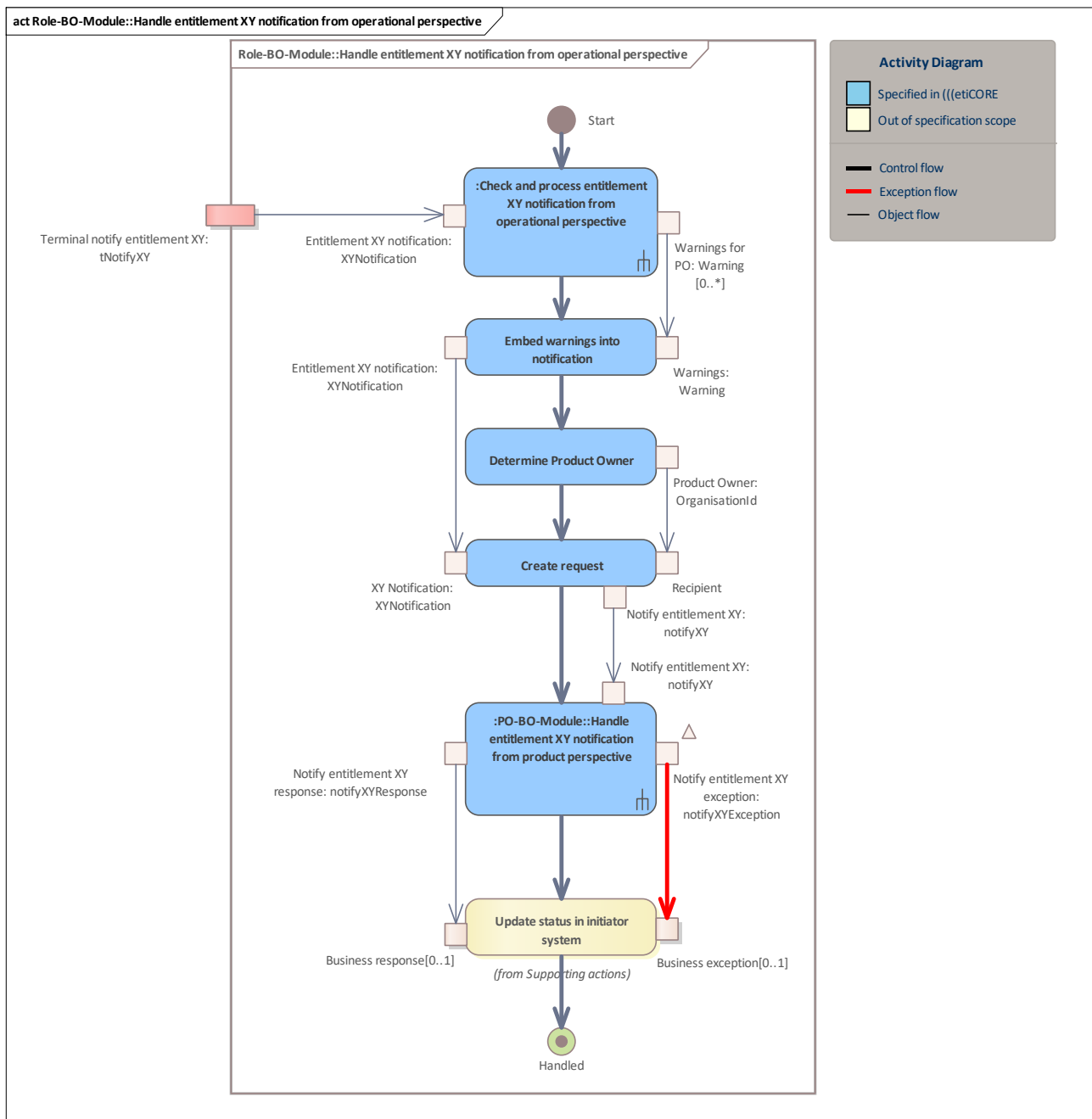


Figure 28: Role-BO-Module::Handle entitlement XY notification from operational perspective

### 3.5.3 Check and process entitlement XY notification from operational perspective

This activity performs the system internal processing of an entitlement-based notification coming from a terminal. All necessary monitoring checks are performed, divided into general checks and notification-specific checks.

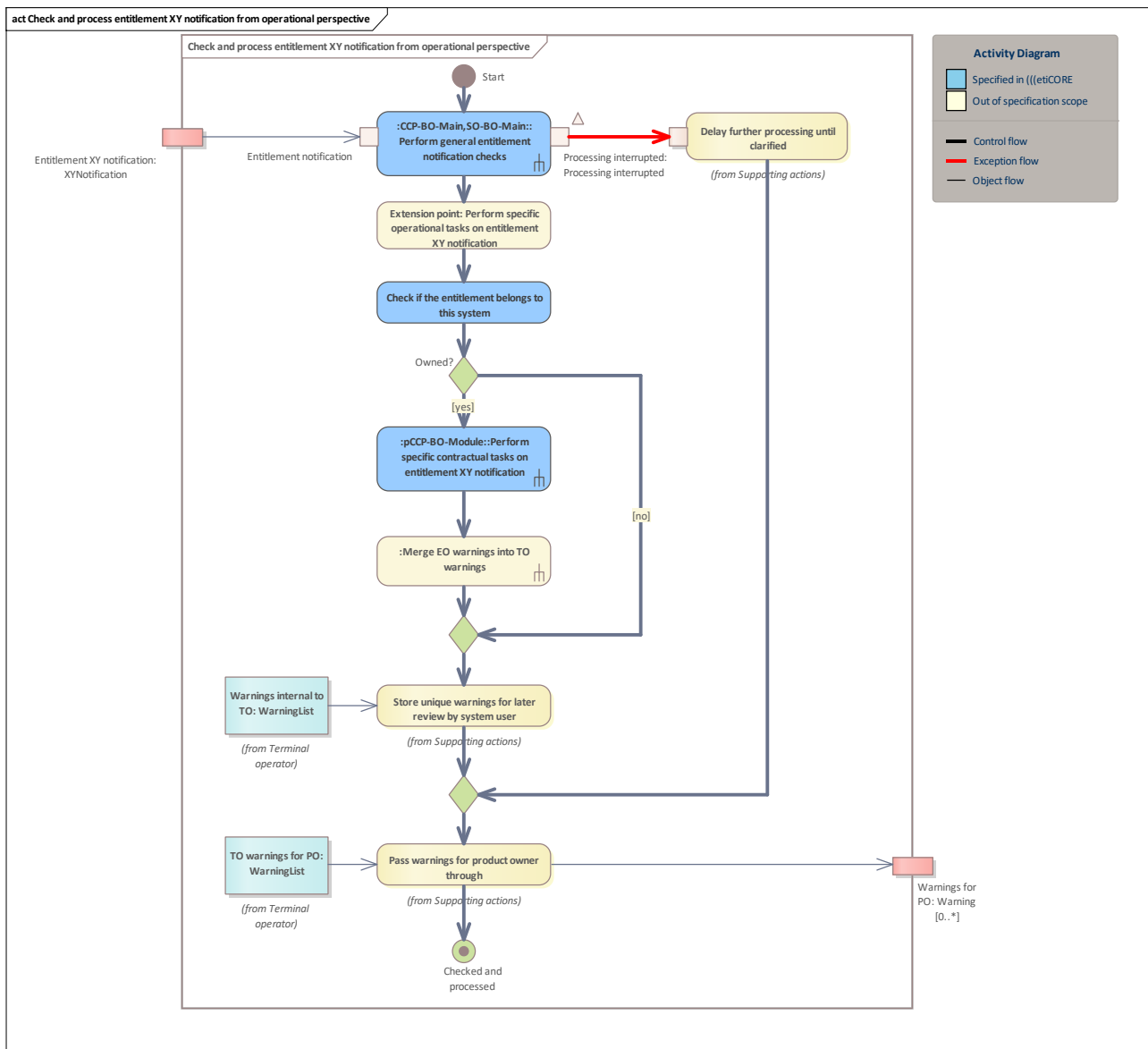


Figure 29: Check and process entitlement XY notification from operational perspective

### 3.5.3.1 Extension point: Perform specific operational tasks on entitlement XY notification

Log SAM action counter value / SAM entitlement issuance counter value / product issuance counter value usage, etc.

### 3.5.4 Role-BO-Module::Handle entitlement XY aborted notification from operational perspective

The back-office system belonging to the terminal handles a notification about an abortion during entitlement XY from an operational perspective.

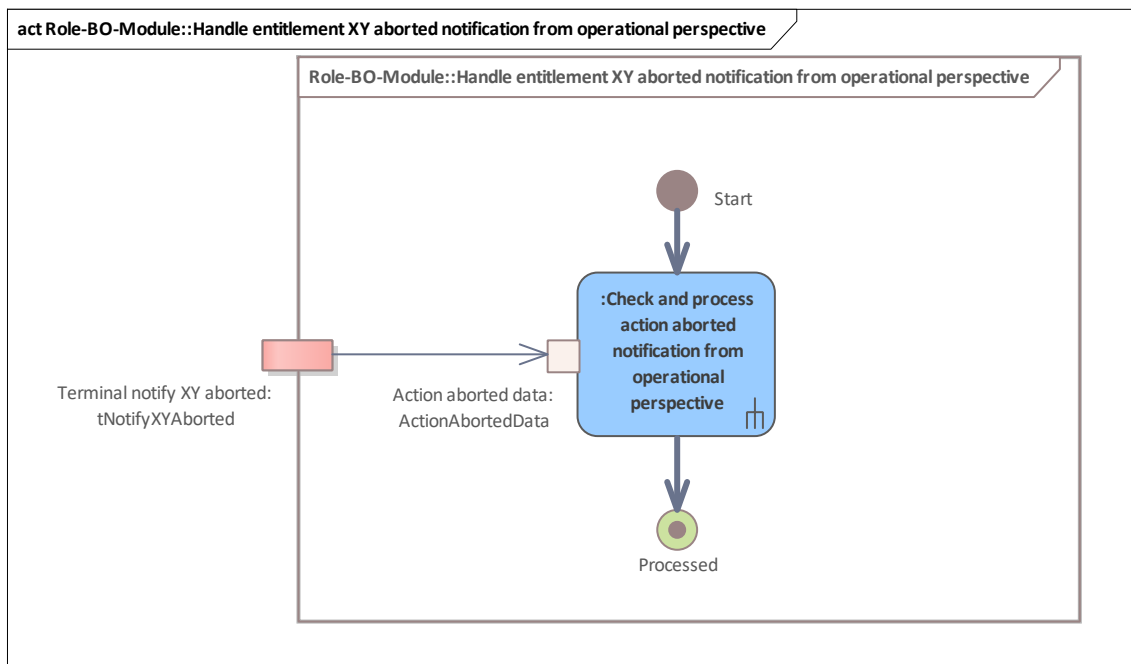


Figure 30: Role-BO-Module::Handle entitlement XY aborted notification from operational perspective

## 3.6 Handle application XY notification from operational perspective

See [Handle application XY notification from operational perspective](#)

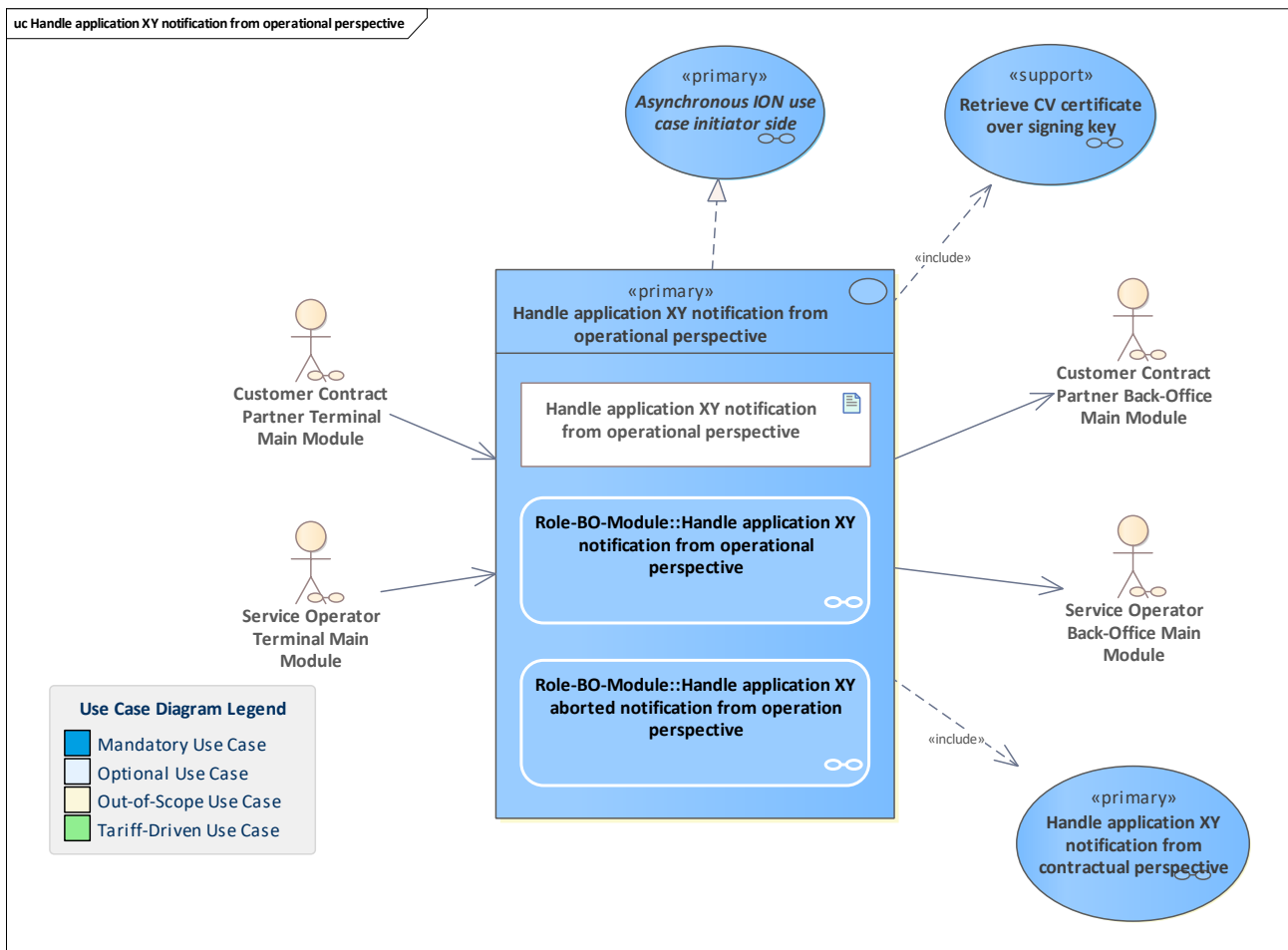


Figure 31: Handle application XY notification from operational perspective

### 3.6.1 Handle application XY notification from operational perspective

A back-office system belonging to a terminal that executed a UM action with an application processes the notification about that action execution.

The processing is done from the operational perspective, focusing on the terminal-side of the action execution, i.e. logging the used SAM counter values.

Depending on the use case and the ownership of the application the action was executed on, other back-office systems are informed about the action execution.

### 3.6.2 Role-BO-Module::Handle application XY notification from operational perspective

See [Handle application XY notification from operational perspective](#)

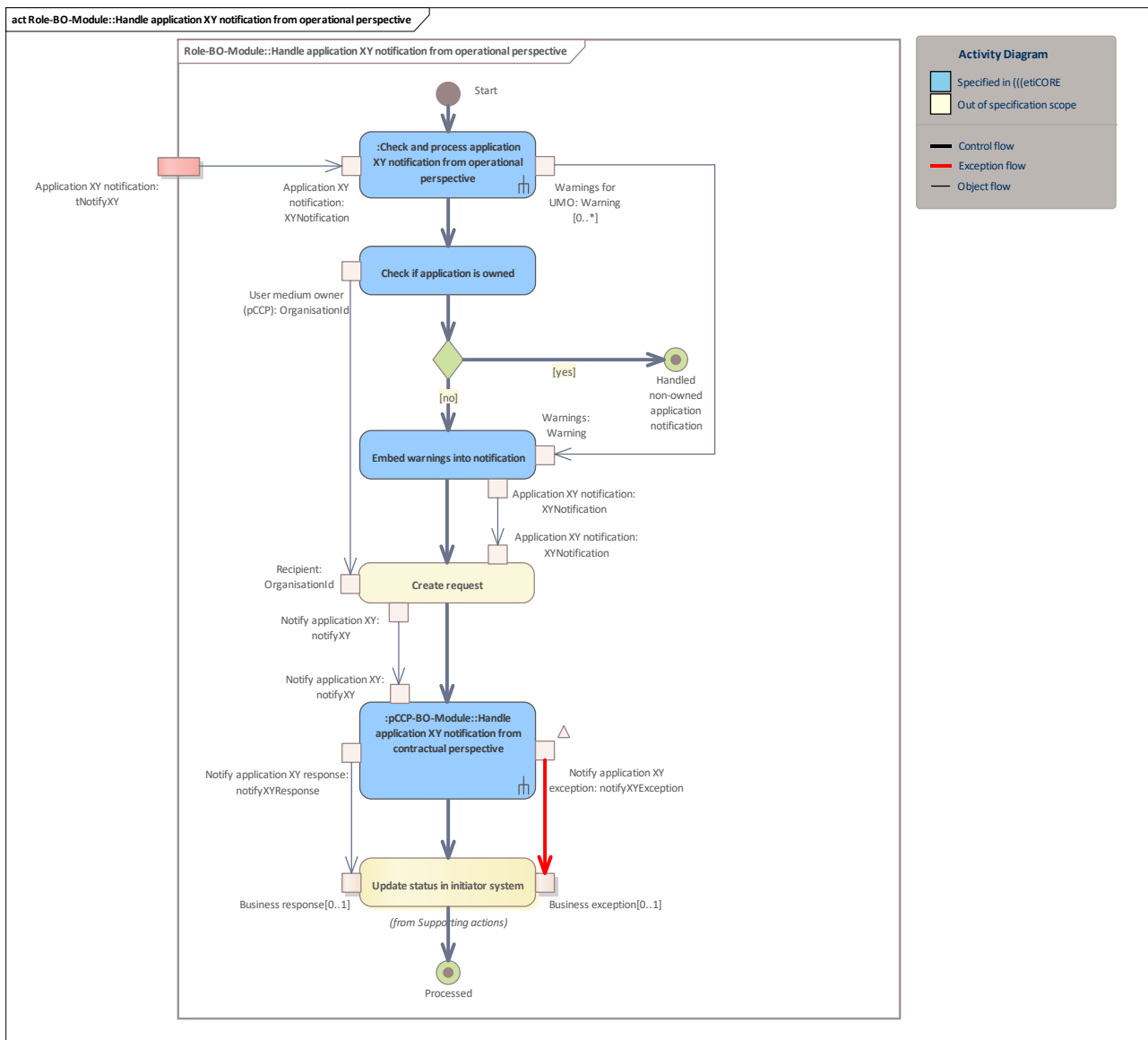


Figure 32: Role-BO-Module::Handle application XY notification from operational perspective

### 3.6.3 Check and process application XY notification from operational perspective

This activity performs the system internal processing of an application-based notification coming from a terminal. All necessary monitoring checks are performed, divided into general checks and notification specific checks.

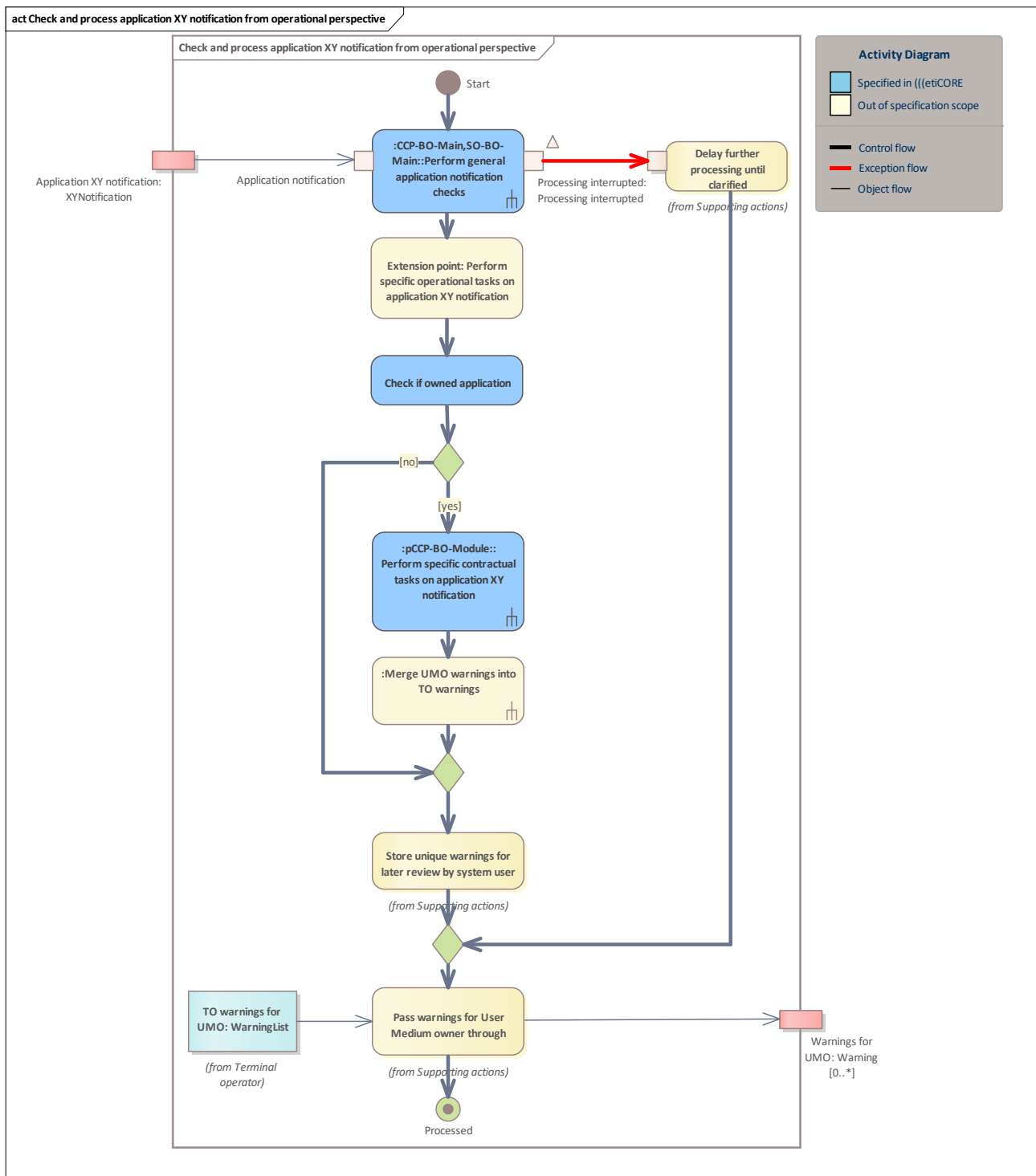


Figure 33: Check and process application XY notification from operational perspective

### 3.6.3.1 Extension point: Perform specific operational tasks on application XY notification

Log SAM action counter value, etc.

### 3.6.4 Role-BO-Module::Handle application XY aborted notification from operation perspective

The back-office system belonging to the terminal handles a notification about an abortion during application XY from an operational perspective.

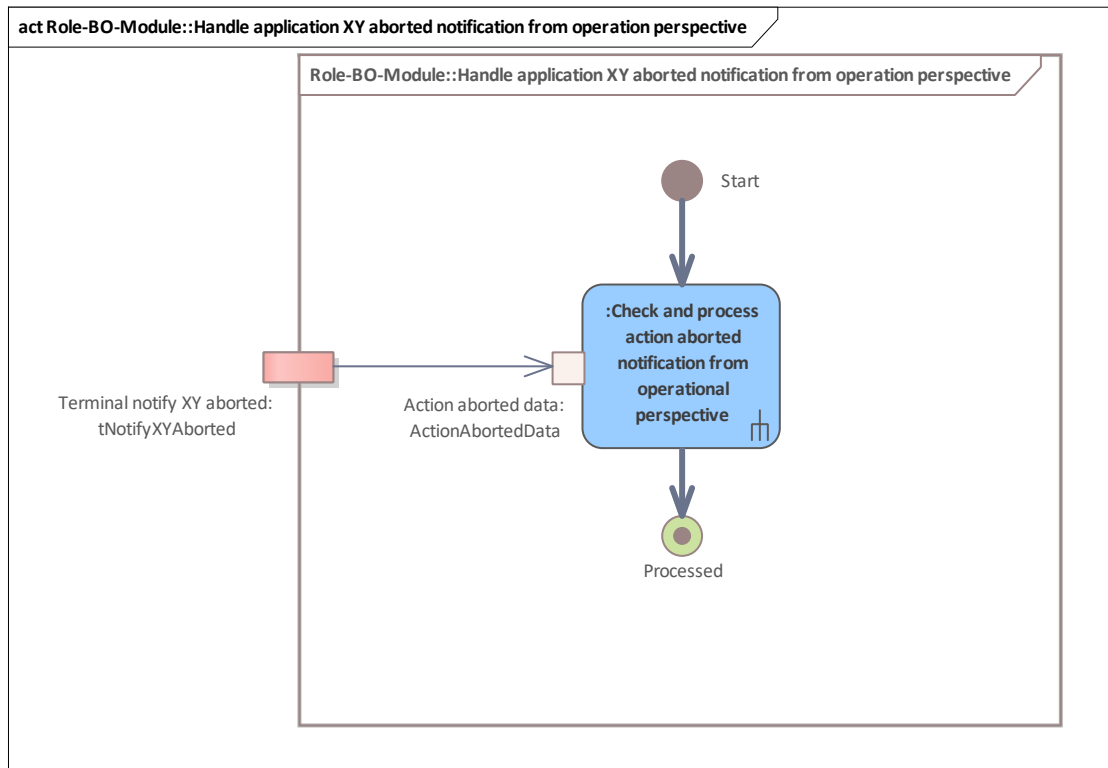


Figure 34: Role-BO-Module::Handle application XY aborted notification from operation perspective

## 3.7 Handle entitlement XY notification from product perspective

See [Handle entitlement XY notification from product perspective](#)



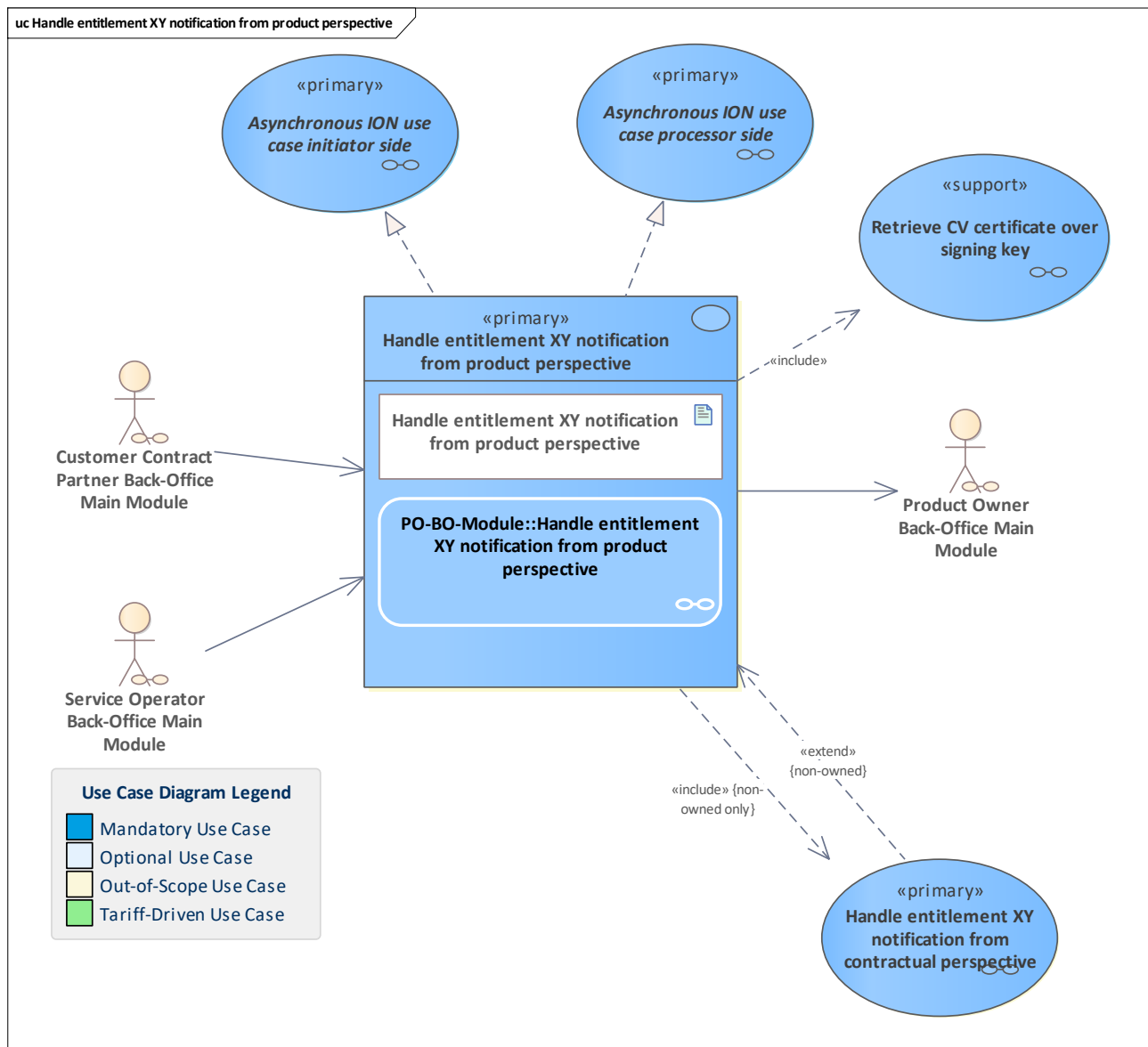


Figure 35: Handle entitlement XY notification from product perspective

### 3.7.1 Handle entitlement XY notification from product perspective

A PO system processes a notification about an entitlement action executed on an entitlement that is an instance of an owned product.  
The processing is done from the product perspective, focusing on the entitlement lifecycle and tariff aspects.

### 3.7.2 PO-BO-Module::Handle entitlement XY notification from product perspective

See [Handle entitlement XY notification from product perspective](#)

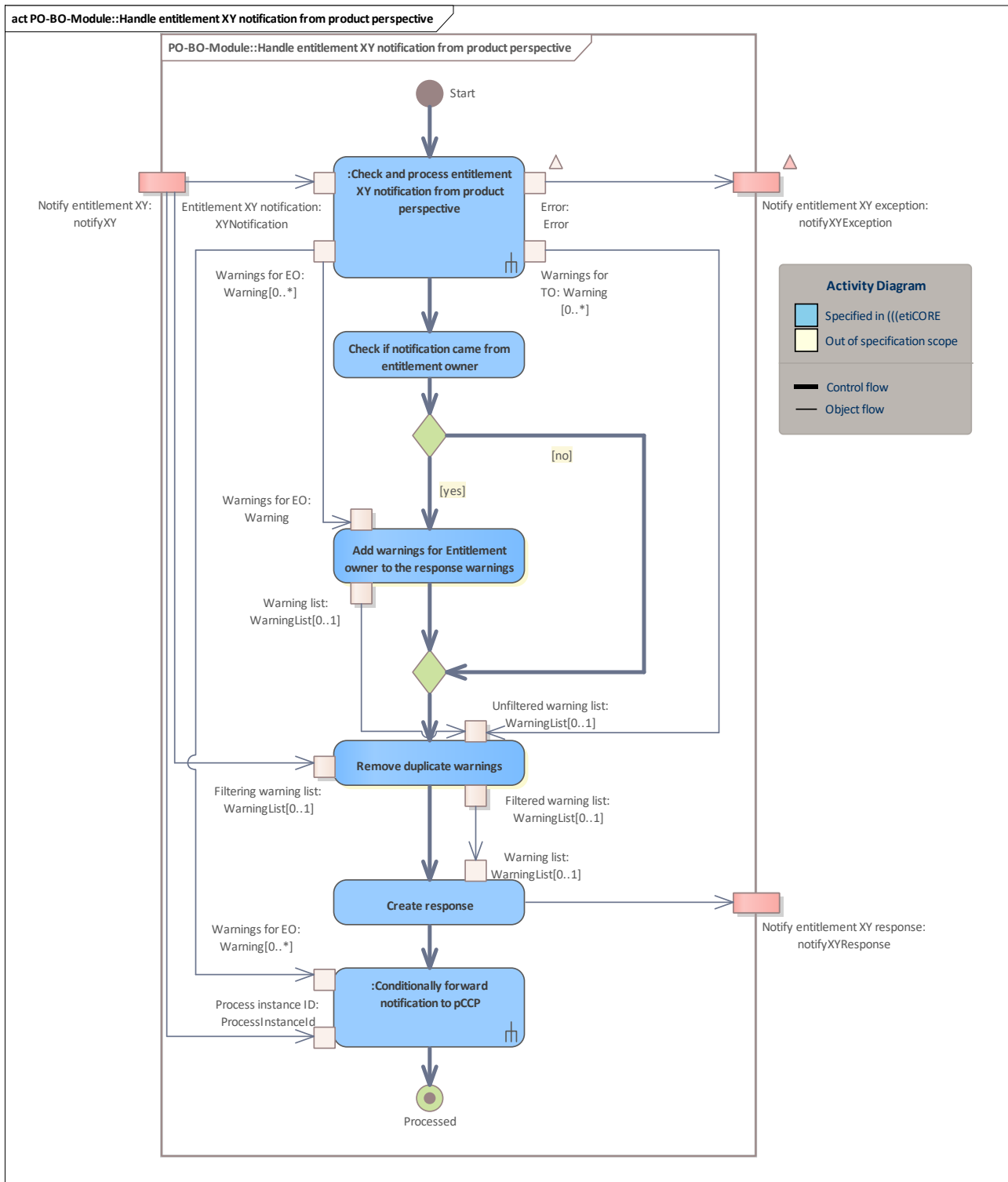


Figure 36: PO-BO-Module::Handle entitlement XY notification from product perspective

### 3.7.3 Check and process entitlement XY notification from product perspective

This activity performs the system internal processing of an entitlement-based notification coming from the terminal operator system (SO or CCP). All necessary monitoring checks are performed, divided into general checks and notification specific checks.

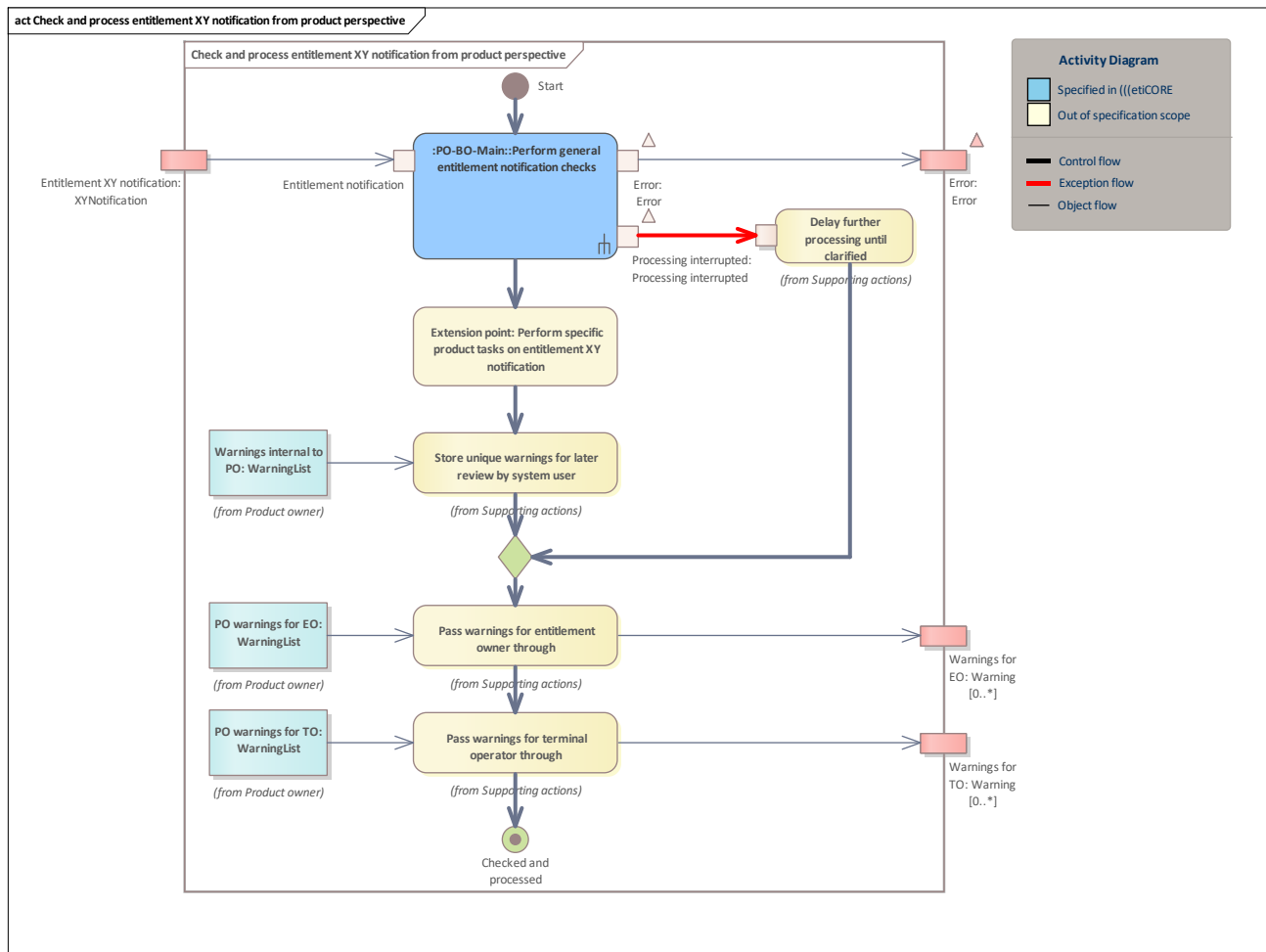


Figure 37: Check and process entitlement XY notification from product perspective

### 3.7.4 Conditionally forward notification to pCCP

Activity that is used if the sender of the notification was not owner of the entitlement. In this case, the notification is forwarded to the primary customer contract partner.

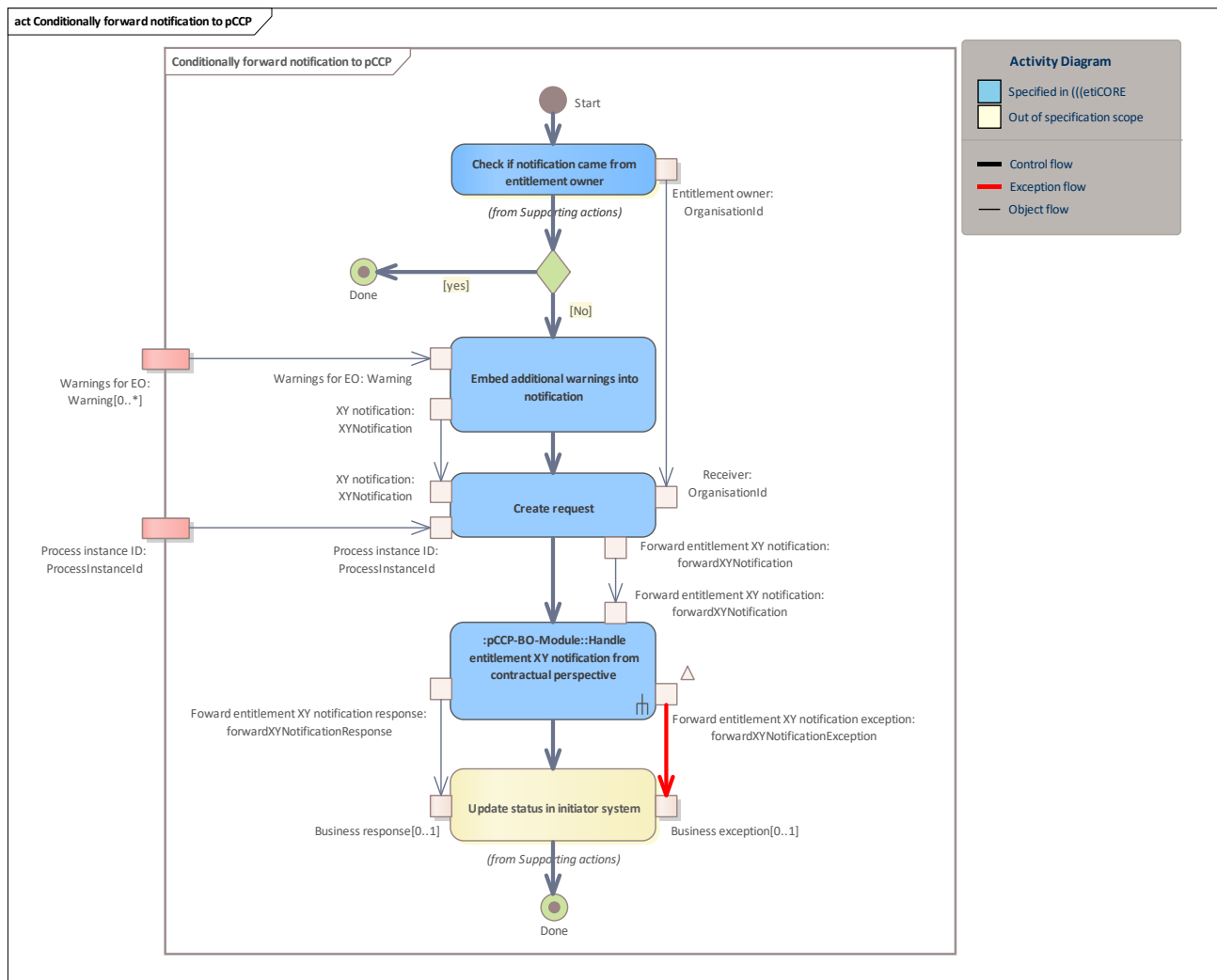


Figure 38: Conditionally forward notification to pCCP

### 3.7.4.1 Embed additional warnings into notification

Embed the given warnings into the notification currently being handled.

## 3.8 Handle entitlement XY notification from contractual perspective

See [Handle entitlement XY notification from contractual perspective](#)

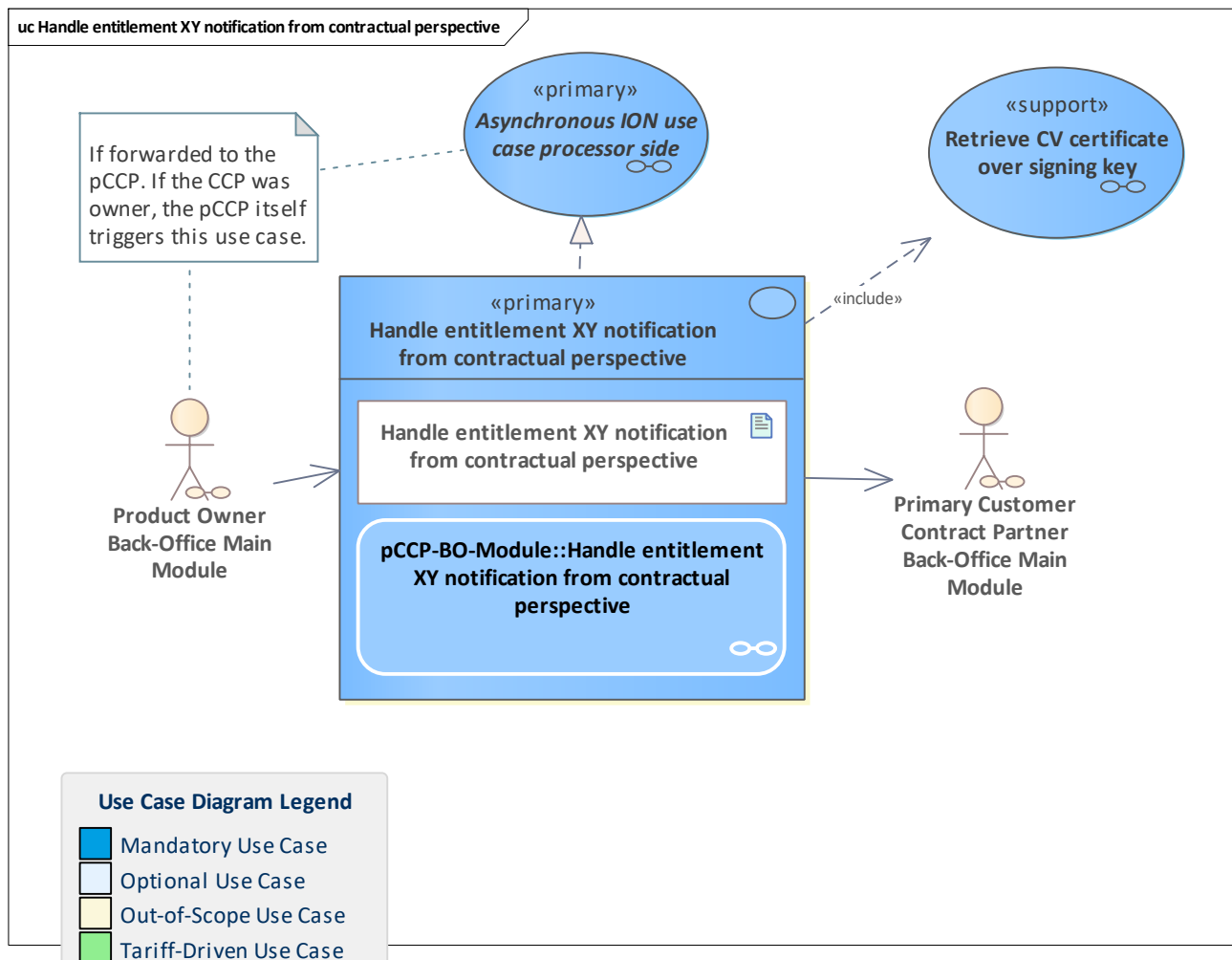


Figure 39: Handle entitlement XY notification from contractual perspective

### 3.8.1 Handle entitlement XY notification from contractual perspective

A CCP system processes a notification about an action executed on an owned entitlement. The processing is done from the contractual perspective focusing on the entitlement lifecycle and payment aspects.

This use case has two entry points:

- [pCCP-BO-Module::Handle entitlement XY notification from contractual perspective](#)  
This entry point is used in the non-owned scenarios
- [pCCP-BO-Module::Perform specific contractual tasks on entitlement XY notification](#)  
This entry point is used in the owned scenarios

The first includes the latter, in which most of the processing is done.

### 3.8.2 pCCP-BO-Module::Handle entitlement XY notification from contractual perspective

See [Handle entitlement XY notification from contractual perspective](#)

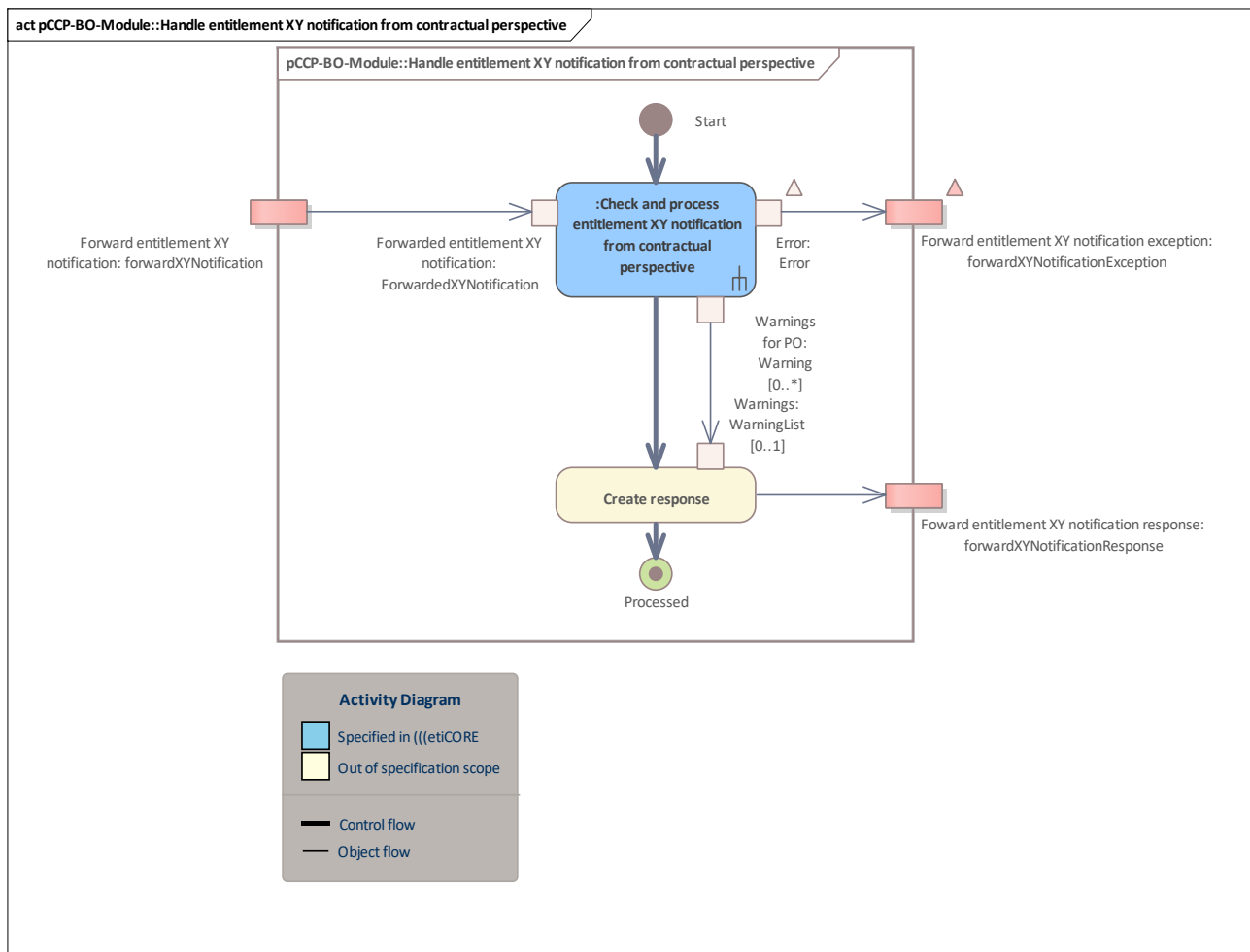


Figure 40: pCCP-BO-Module::Handle entitlement XY notification from contractual perspective

### 3.8.3 Check and process entitlement XY notification from contractual perspective

This activity performs the system internal processing of an entitlement-based notification either forwarded by the PO system or triggered directly after [Handle entitlement XY notification from operational perspective](#), if the current actor is the owner of the entitlement . All necessary monitoring checks are performed, divided into general checks and notification specific checks.

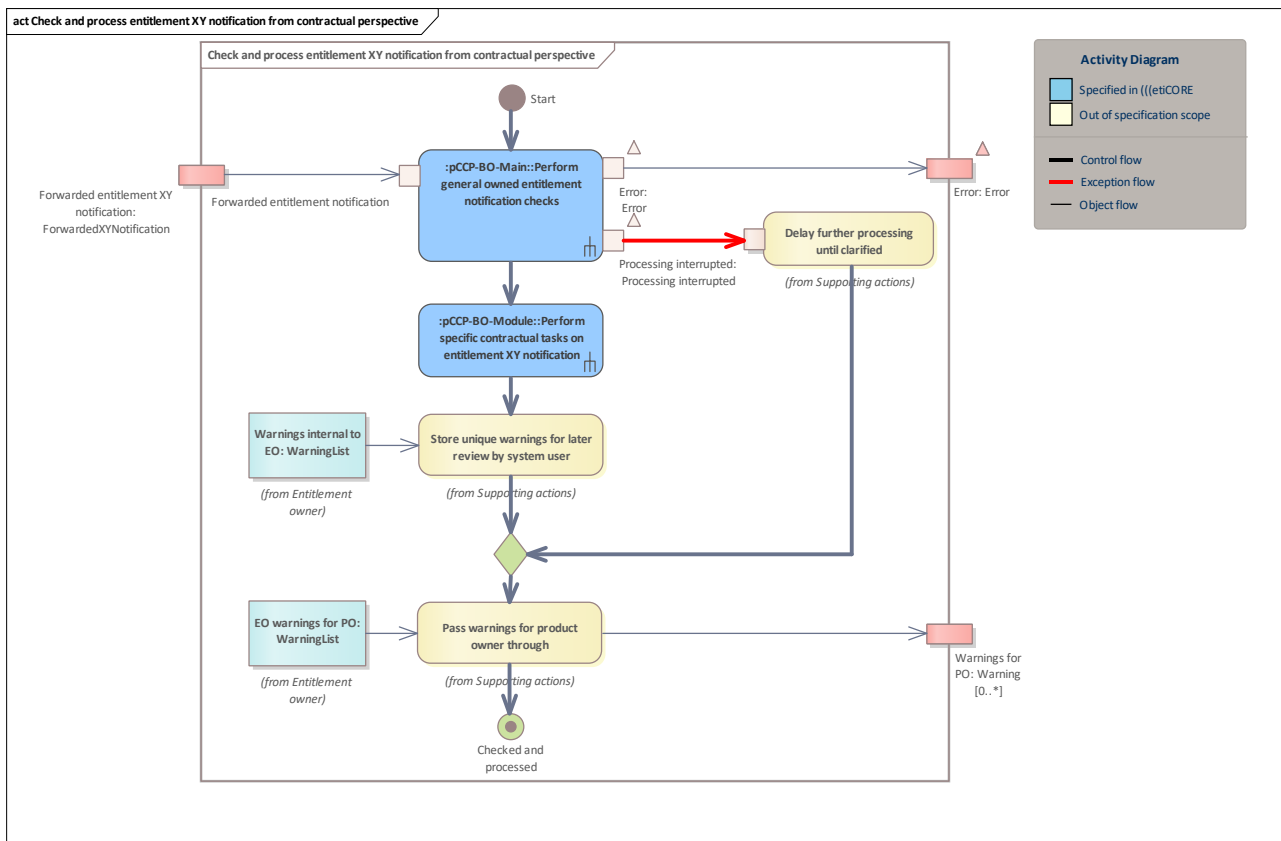


Figure 41: Check and process entitlement XY notification from contractual perspective

### 3.8.4 pCCP-BO-Module::Perform specific contractual tasks on entitlement XY notification

See [Handle entitlement XY notification from contractual perspective](#)

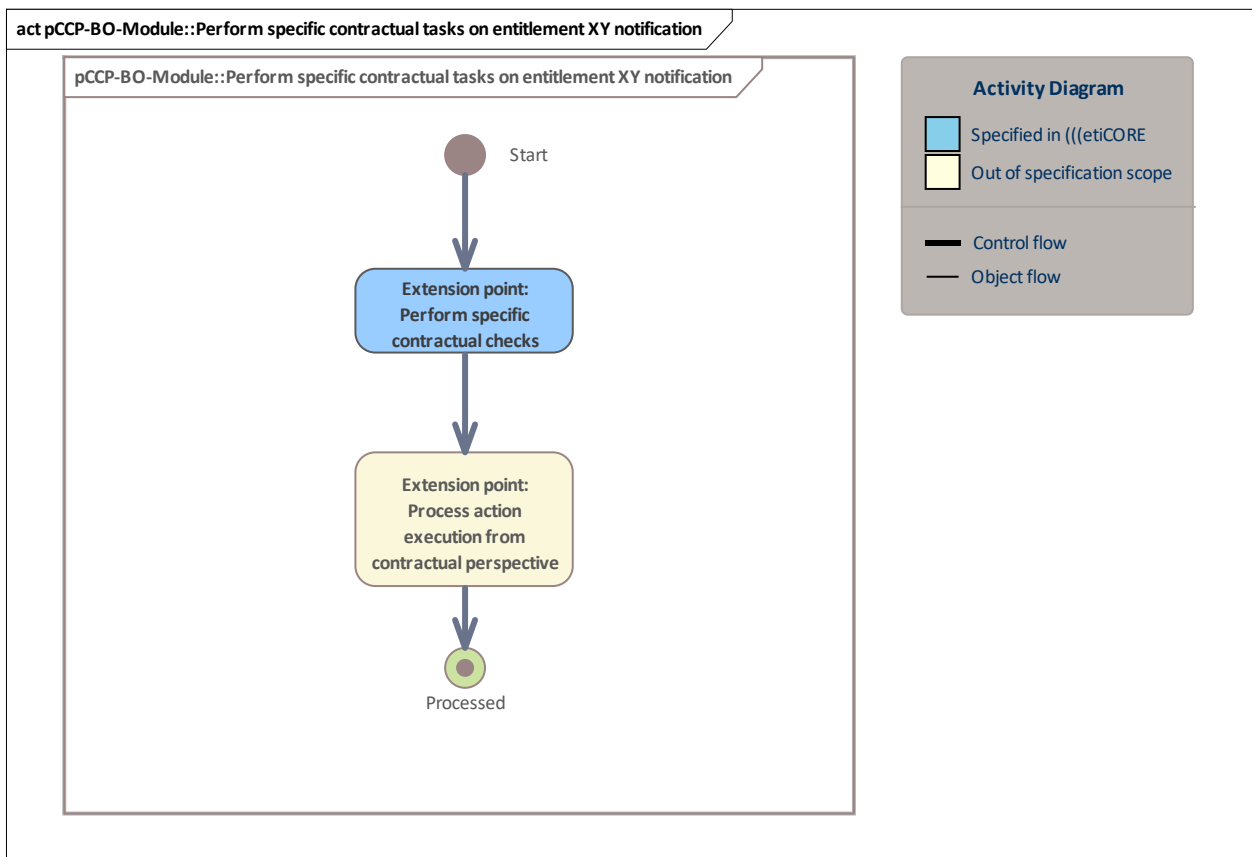


Figure 42: pCCP-BO-Module::Perform specific contractual tasks on entitlement XY notification

### 3.8.4.1 Extension point: Process action execution from contractual perspective

Update sales register, update entitlement state, etc.

## 3.9 Handle application XY notification from contractual perspective

See [Handle application XY notification from contractual perspective](#)



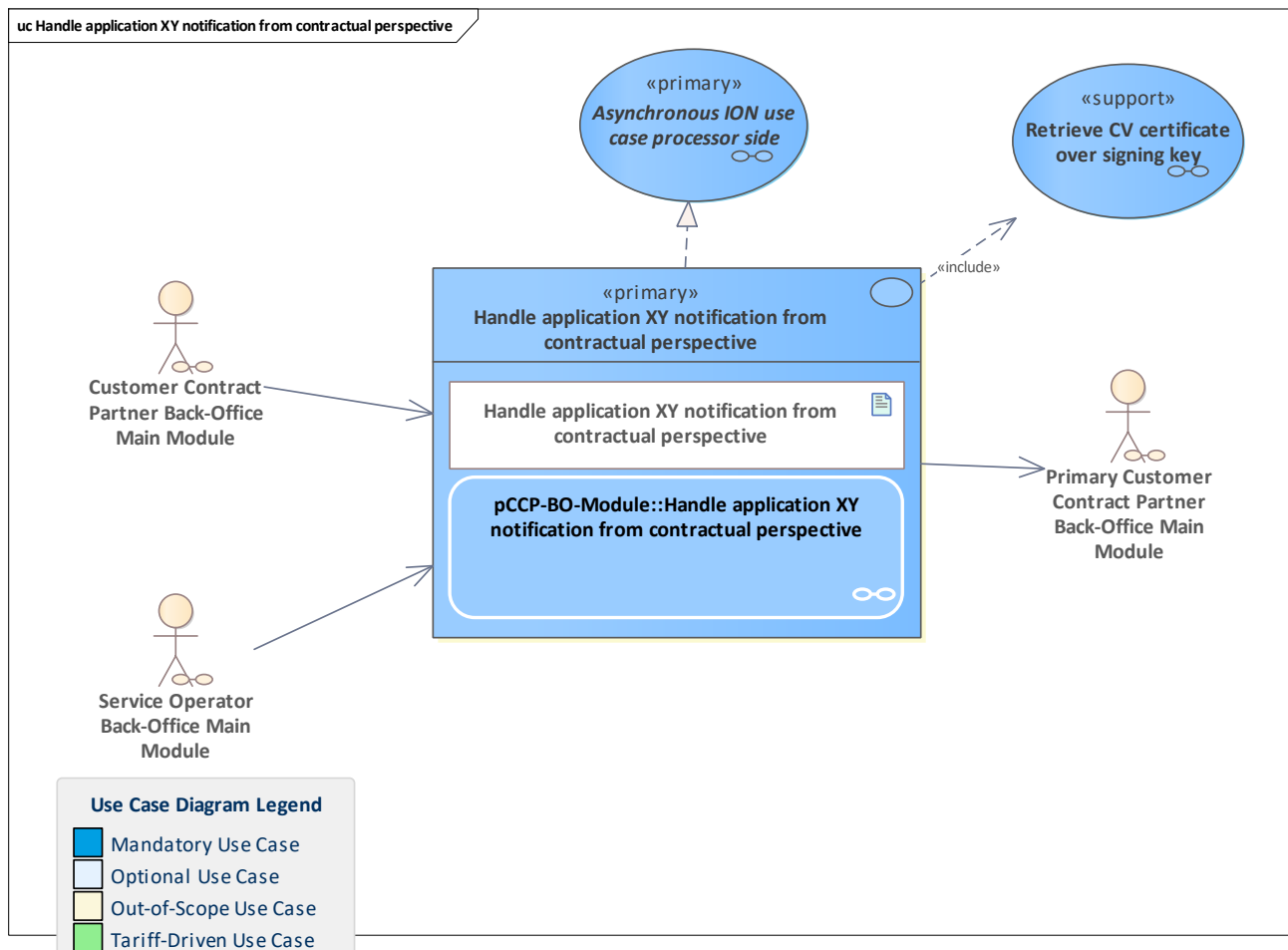


Figure 43: Handle application XY notification from contractual perspective

### 3.9.1 Handle application XY notification from contractual perspective

A CCP system processes a notification about an action executed on an owned application. The processing is done from the contractual perspective focusing on the application lifecycle aspects.

This use case has two entry points:

- [pCCP-BO-Module::Handle application XY notification from contractual perspective](#)  
This entry point is used in the non-owned scenarios
- [pCCP-BO-Module::Perform specific contractual tasks on application XY notification](#)  
This entry point is used in the owned scenarios

The first includes the latter, in which most of the processing is done.

### 3.9.2 pCCP-BO-Module::Handle application XY notification from contractual perspective

See [Handle application XY notification from contractual perspective](#)

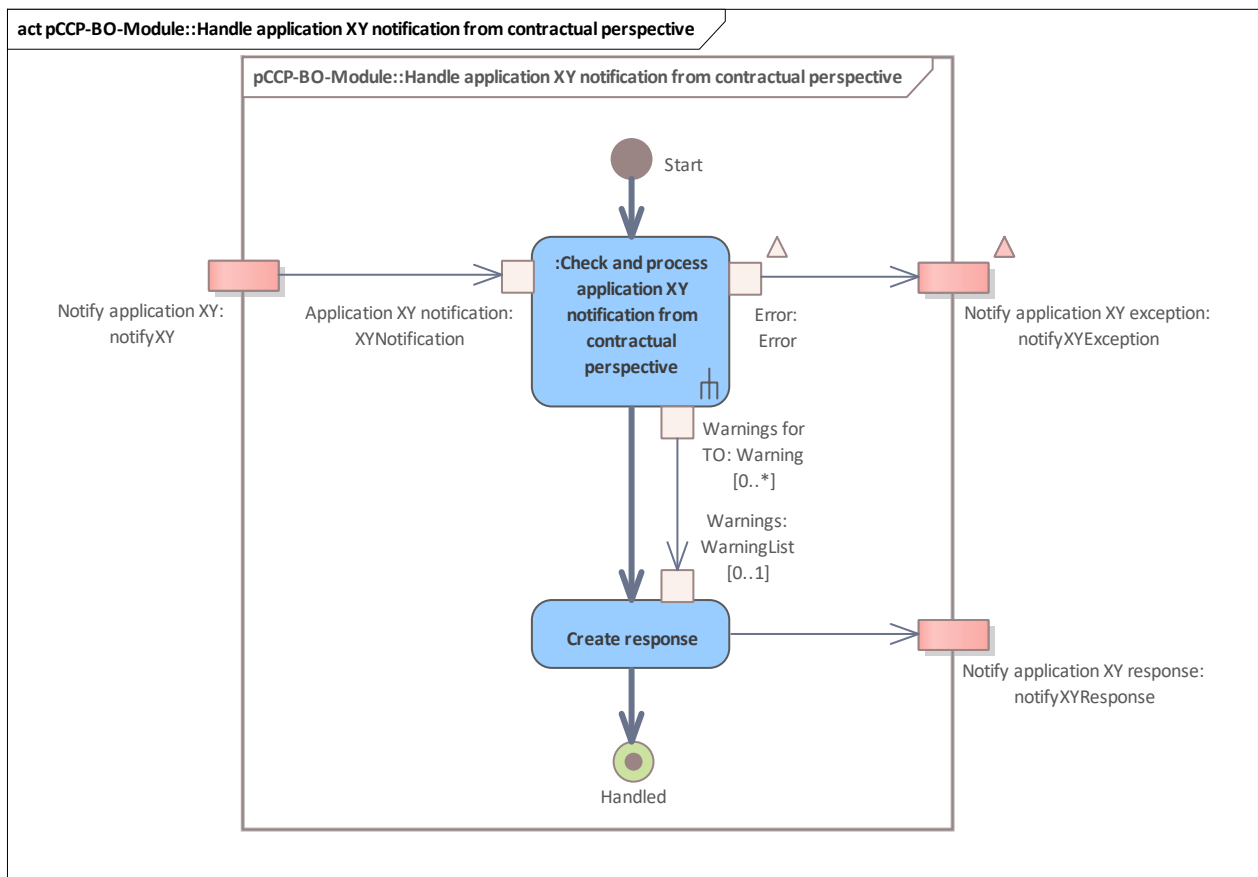


Figure 44: pCCP-BO-Module::Handle application XY notification from contractual perspective

### 3.9.3 Check and process application XY notification from contractual perspective

This activity performs the system internal processing of an application based notification either sent from another terminal operator system (SO or sCCP) or triggered directly after the executed use case [Handle application XY notification from operational perspective](#), if the current actor is the owner of the application instance. All necessary monitoring checks are performed, divided into general checks and notification specific checks.

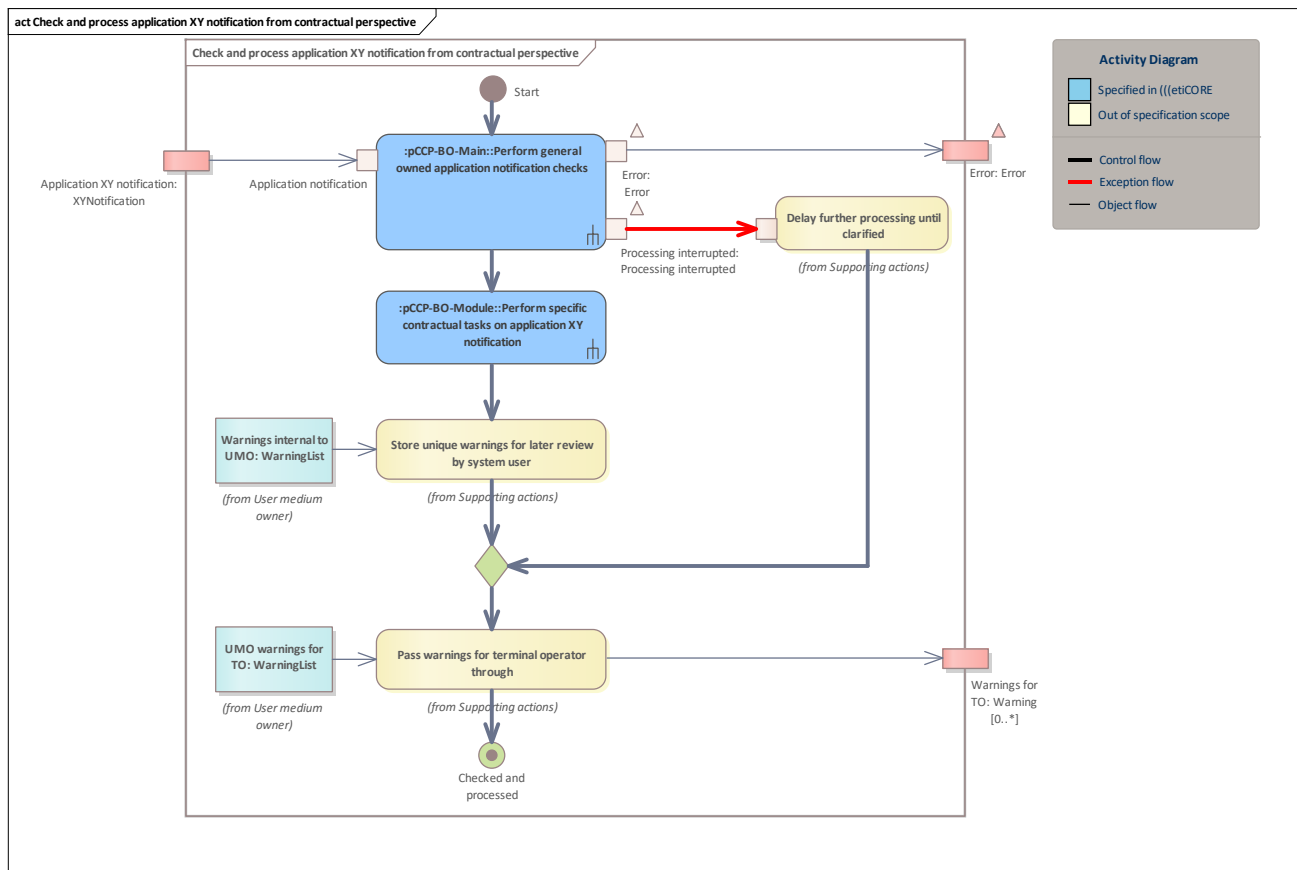


Figure 45: Check and process application XY notification from contractual perspective

### 3.9.4 pCCP-BO-Module::Perform specific contractual tasks on application XY notification

See [Handle application XY notification from contractual perspective](#)

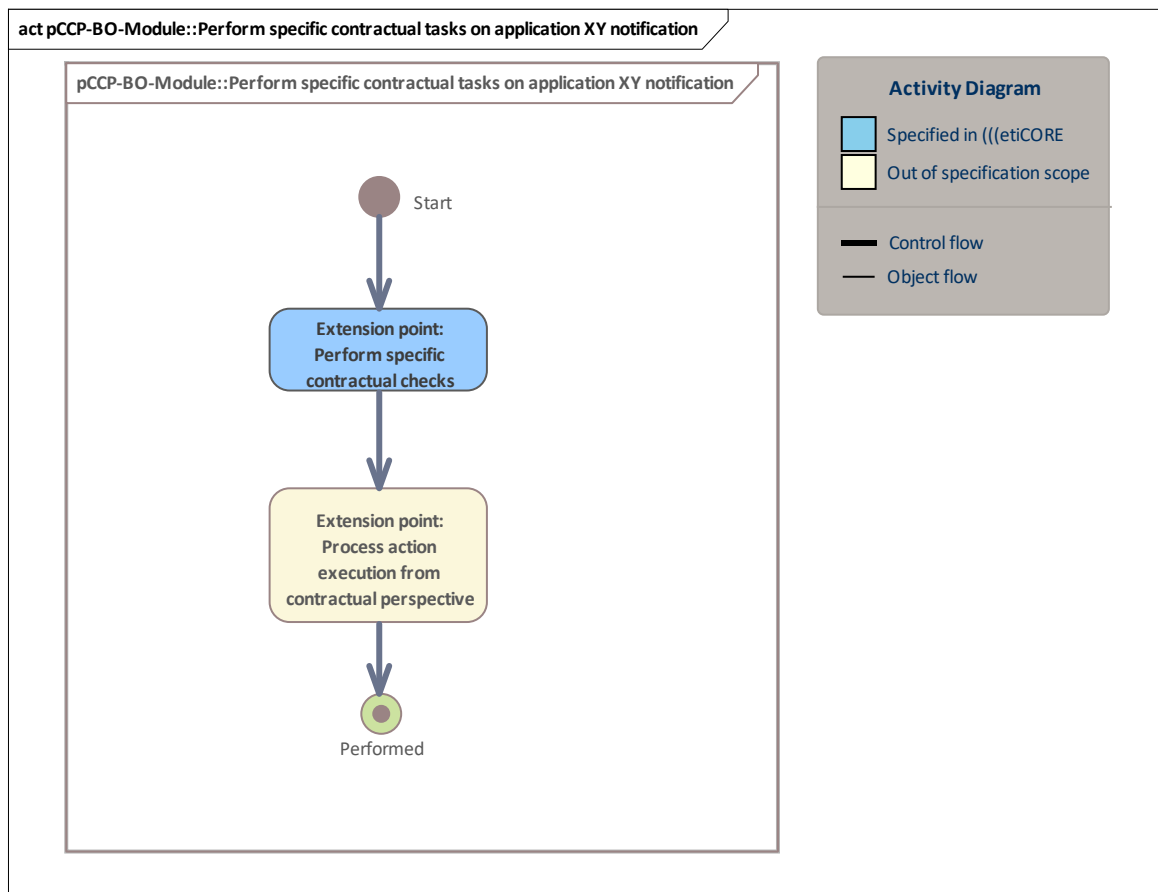


Figure 46: pCCP-BO-Module::Perform specific contractual tasks on application XY notification

## 4 Verifying Lists Updated via Increments

Incremental lists are used for large lists containing thousands of entries. These lists occur in the hotlist service for lists of hotlisted application instances and entitlements, as well as in the action list service when fetching the newest action list.

Instead of fetching a full list, an incremental list can be used. This list only contains the differences between a specified last list and the current one.

These differences have to be merged into the current inventory. To verify that the updated inventory is equal to the inventory of a full list, a certain checksum algorithm is used.

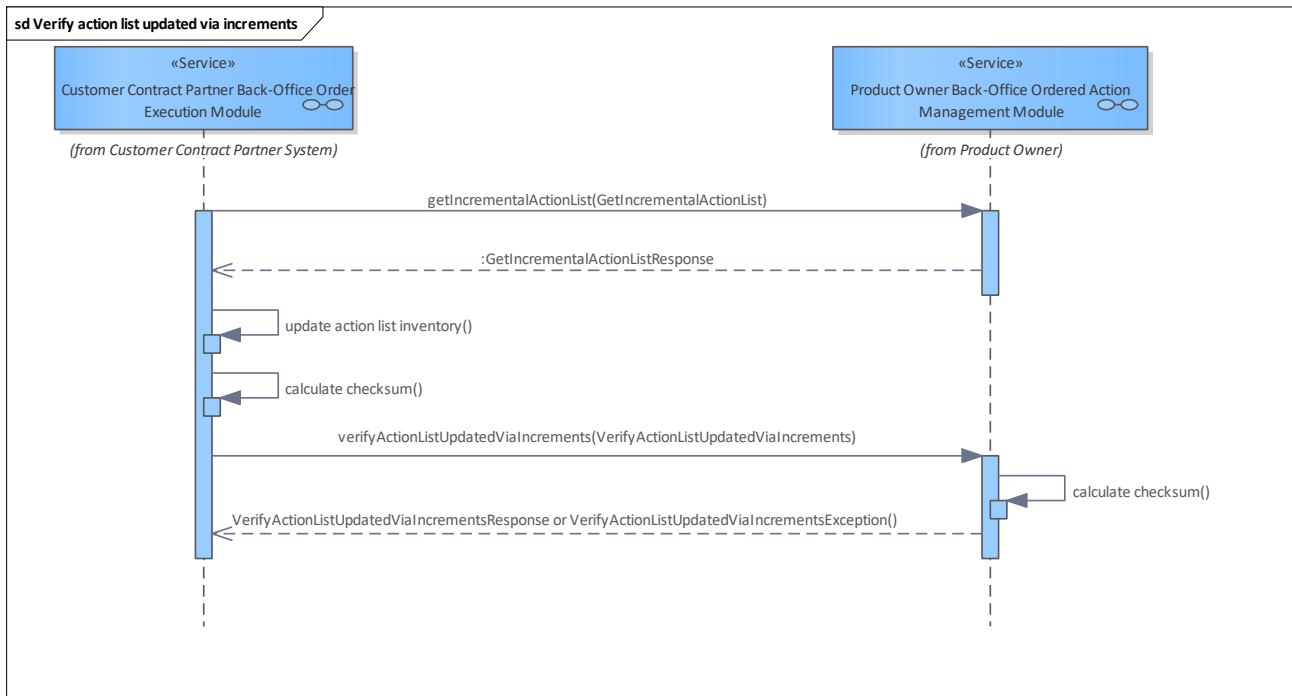


Figure 47: Verify action list updated via increments



Figure 48: Verify application hotlist updated via increments



Figure 49: Verify entitlement hotlist updated via increments

## 4.1 Checksum calculation for hotlist and action list verification

The actual calculation of the checksum differs per list and is given in the corresponding use cases. Here, we describe some aspects that apply to all of them.

The data objects are encoded according to the distinguished encoding rules (DER) per the corresponding ASN.1 schemata including tag and length.

The hash algorithm to use is SHA-256.

We always sort lexicographically in ascending order.

Pseudo code to calculate the hash value:

hash( concatenate( sort( [ serialise(entry) for entry in entries ] ) ) )

## 4.2 Example calculation for an action list inventory

Action list inventory (reduced to the parts relevant to hashing) to verify:

```

[
  {
    "entitlementUnblockingActionListEntry": {
      "actionListEntryBaseInformation": {
        "orderId": {
          "orderNumber": 108,
          "orderingCcp": 123
        }
      }
    }
  },
  ...
]
  
```

```
{
  "entitlementIssuanceActionListEntry": {
    "actionListEntryBaseInformation": {
      "orderId": {
        "orderNumber": 109,
        "orderingCcp": 123
      }
    }
  },
  "entitlementTerminationActionListEntry": {
    "actionListEntryBaseInformation": {
      "orderId": {
        "orderNumber": 111,
        "orderingCcp": 123
      }
    }
  },
  "entitlementBlockingActionListEntry": {
    "actionListEntryBaseInformation": {
      "orderId": {
        "orderNumber": 110,
        "orderingCcp": 123
      }
    }
  }
}
```

Intermediate result after serialisation:

```
[0x4D017B5A016C,
0x4D017B5A016D,
0x4D017B5A016F,
0x4D017B5A016E]
```

Intermediate result after sorting:

```
[0x4D017B5A016C,
0x4D017B5A016D,
0x4D017B5A016E,
0x4D017B5A016F]
```

Final result after applying SHA-256:

```
0x1376FF1481CB9B73DDADDC5F4588478A506EE4129AA345B2D408E37DF1FB51E8
```

## 4.3 Example calculation for an application hotlist inventory

Application hotlist inventory to verify:

```
[
  {
    "applicationInstanceId": {
      "registrationAuthorityId": 10004,
```

```
        "subjectRole": "0x1F",
        "subjectNumber": "0x00000004"
    },
    "applicationTransitionCounter": 0,
    "applicationBlockingMode": 10
},
{
    "applicationInstanceId": {
        "registrationAuthorityId": 10004,
        "subjectRole": "0x1B",
        "subjectNumber": "0x00000002"
    },
    "applicationTransitionCounter": 6,
    "applicationBlockingMode": 0
},
{
    "applicationInstanceId": {
        "registrationAuthorityId": 10004,
        "subjectRole": "0x1B",
        "subjectNumber": "0x00000001"
    },
    "applicationTransitionCounter": 0,
    "applicationBlockingMode": 0
}
]
```

Intermediate result after serialisation:

```
[0x640D4D02271480011F82040000000458010002010A,
0x640D4D02271480011B820400000002580106020100,
0x640C4D02271480011B8203000001580100020100]
```

Intermediate result after sorting:

```
[0x640C4D02271480011B8203000001580100020100,
0x640D4D02271480011B820400000002580106020100,
0x640D4D02271480011F82040000000458010002010A]
```

Final result after applying SHA-256:

```
0x98152207B15DE600F0A98974CA510DDB6A3034E661BFE4287BF808128C61C6E2
```

## 4.4 Example calculation for an entitlement hotlist inventory

Entitlement hotlist inventory to verify:

```
[
    {
        "entitlementBlockingMode": 0,
        "entitlementId": {
            "ccpOrgId": 348,
            "entitlementIssuanceCounter": 1005,
            "samId": {
                "registrationAuthorityId": 2713,
                "subjectNumber": "0x00000002",
                "subjectRole": "0x13"
            }
        }
    },
    {
        "entitlementBlockingMode": 0,
        "entitlementId": {
            "ccpOrgId": 348,
            "entitlementIssuanceCounter": 1005,
            "samId": {
                "registrationAuthorityId": 2713,
                "subjectNumber": "0x00000001",
                "subjectRole": "0x13"
            }
        }
    }
]
```



```
"entitlementTransitionCounter": 0
},
{
  "entitlementBlockingMode": 20,
  "entitlementId": {
    "ccpOrgId": 5730,
    "entitlementIssuanceCounter": 120,
    "samId": {
      "registrationAuthorityId": 2713,
      "subjectNumber": "0x00000002",
      "subjectRole": "0x13"
    }
  }
},
"entitlementTransitionCounter": 0
},
{
  "entitlementBlockingMode": 0,
  "entitlementId": {
    "ccpOrgId": 36,
    "entitlementIssuanceCounter": 48905,
    "samId": {
      "registrationAuthorityId": 2713,
      "subjectNumber": "0x00000002",
      "subjectRole": "0x13"
    }
  }
},
"entitlementTransitionCounter": 6
}
]
```

Intermediate result after serialisation:

```
[0x6E174D02015C640D4D020A998001138204000000024B0203ED580100020100,
0x6E164D021662640D4D020A998001138204000000024B0178580100020114,
0x6E174D0124640D4D020A998001138204000000024B0300BF09580106020100]
```

Intermediate result after sorting:

```
[0x6E164D021662640D4D020A998001138204000000024B0178580100020114,
0x6E174D0124640D4D020A998001138204000000024B0300BF09580106020100,
0x6E174D02015C640D4D020A998001138204000000024B0203ED580100020100]
```

Final result after applying SHA-256:

```
0xC58276BE0B189DEAB6350B85588F80EF016153D01002B14F192B85B1FAA38124
```

## 5 Basic Bundle Back-Office System

This functionality bundle contains use cases that all back-office systems of the CCP, SO and PO must implement.

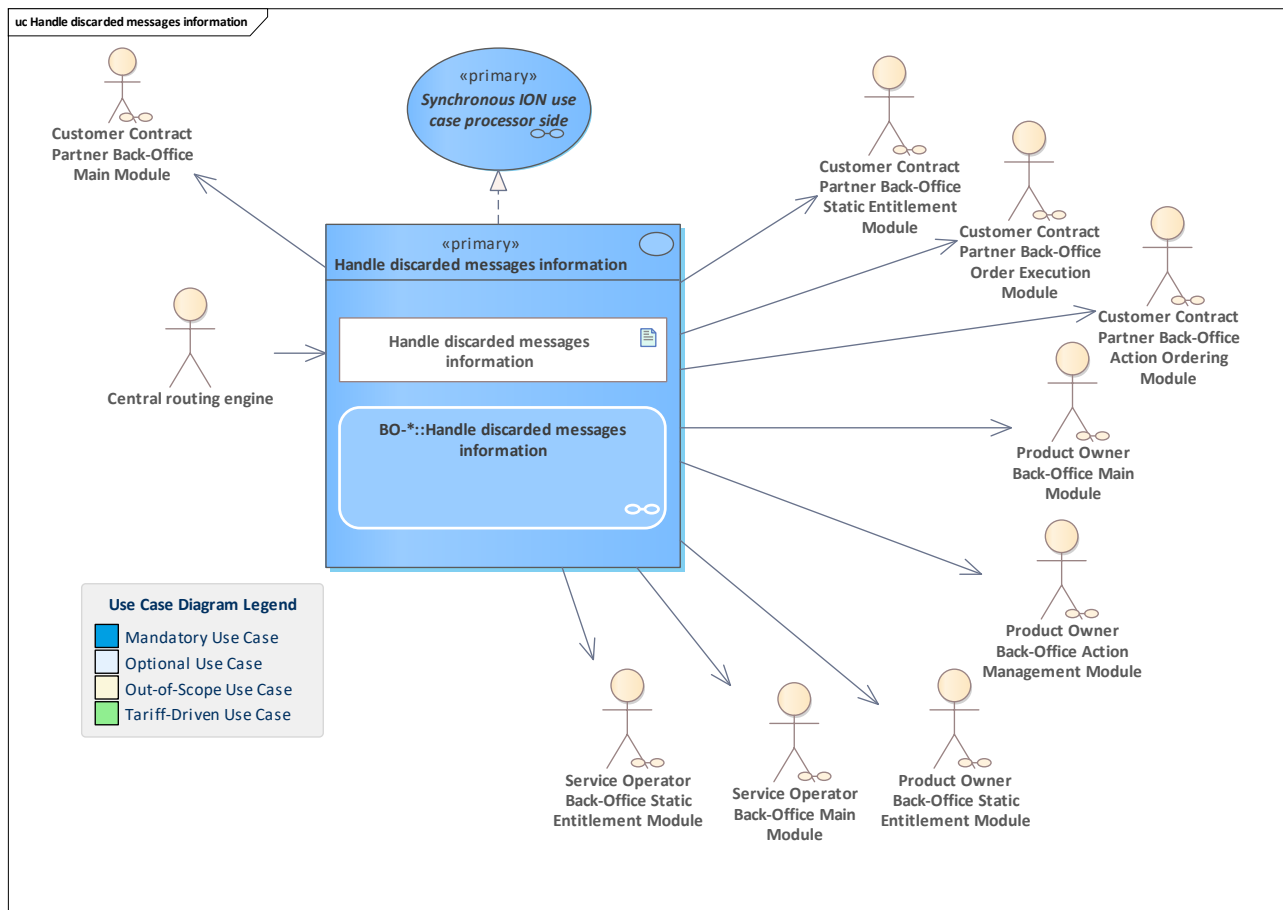
### 5.1 Overview

[Handle discarded messages information](#)  
[Set service as available for a participant](#)  
[Set service as unavailable for a participant](#)  
[Retrieve CV certificate over signing key](#)

[Retrieve and distribute the CA certificate repository](#)  
[Retrieve and distribute the CV certificate revocation list](#)  
[Notify events](#)  
[Handle events notification](#)  
[Demand application hotlisting](#)  
[Determine user medium owner](#)  
[Demand entitlement hotlisting](#)  
[Demand SAM hotlisting](#)  
[Determine SAM owner](#)  
[Revoke application hotlisting demand](#)  
[Revoke entitlement hotlisting demand](#)  
[Retrieve entitlement hotlist](#)  
[Optional: Retrieve entitlement hotlist with product information](#)  
[Optional: Retrieve incremental entitlement hotlist](#)  
[Optional: Verify entitlement hotlist updated via increments](#)  
[Update organisation hotlist inventory](#)  
[Update SAM hotlist inventory](#)  
[Retrieve and distribute organisation list](#)

## 5.2 Use Cases

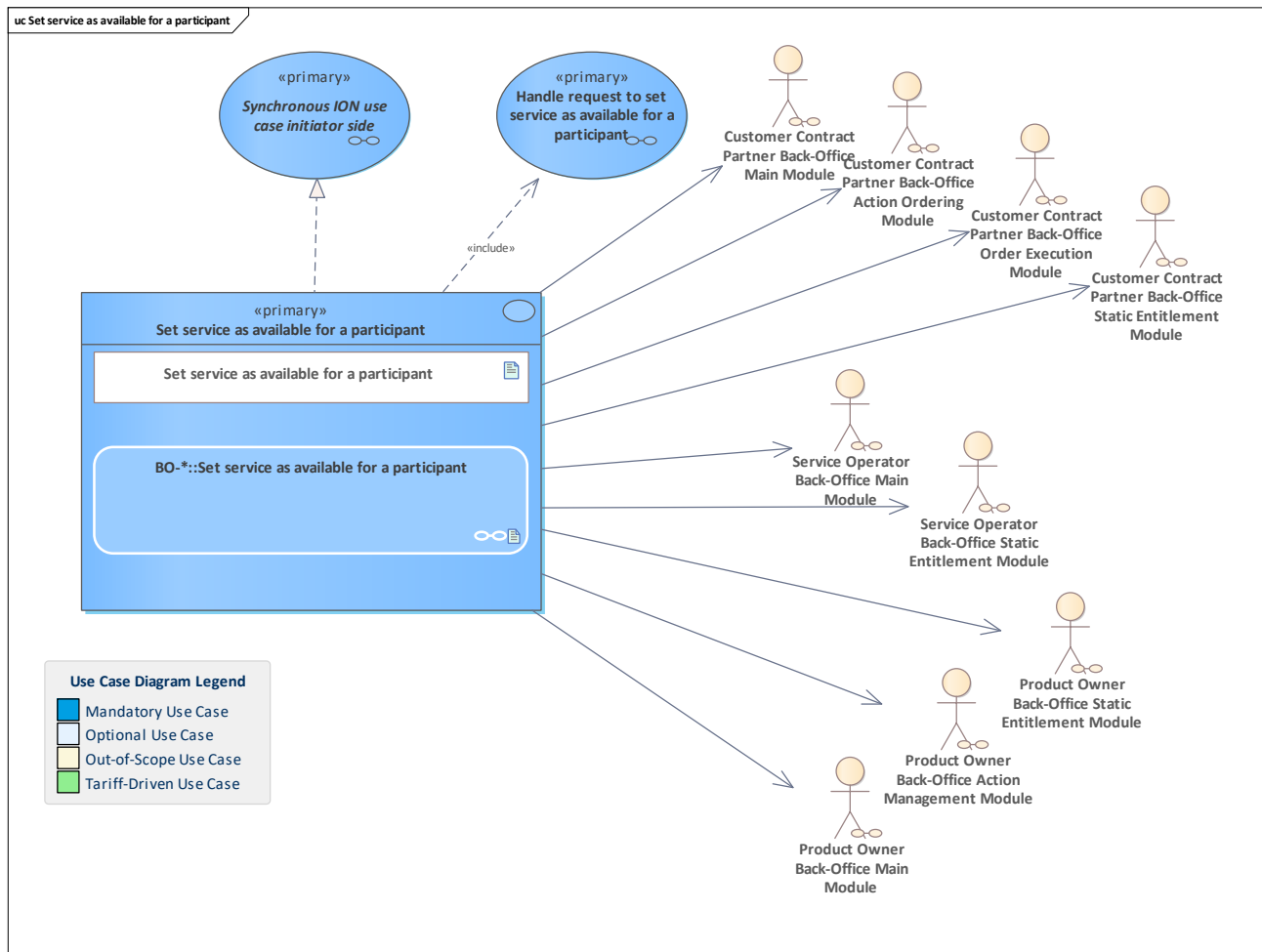
### 5.2.1 Handle discarded messages information





<b>Description</b>	<p>Process information about discarded asynchronous messages which were provided for store &amp; forward but could not be delivered to the specified recipient.</p> <p>The CRE will send message information to a</p> <ul style="list-style-type: none"><li>• <a href="#">Customer Contract Partner System</a></li><li>• <a href="#">Ordering Customer Contract Partner System</a></li><li>• <a href="#">Executing Customer Contract Partner System</a></li><li>• <a href="#">Customer Contract Partner STE System</a></li><li>• <a href="#">Service Operator System</a></li><li>• <a href="#">Service Operator STE System</a></li><li>• <a href="#">Product Owner System</a></li><li>• <a href="#">Product Owner Action Management System</a></li><li>• <a href="#">Product Owner STE System</a></li></ul> <p>These systems and modules work with asynchronous messages which can possibly be discarded.</p>
<b>Initiating Actor</b>	<a href="#">Central routing engine</a>
<b>Reacting Actor</b>	<a href="#">Initiator</a> <a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Action Management Module</a> <a href="#">Customer Contract Partner Back-Office Static Entitlement Module</a> <a href="#">Customer Contract Partner Back-Office Order Execution Module</a> <a href="#">Product Owner Back-Office Static Entitlement Module</a> <a href="#">Service Operator Back-Office Static Entitlement Module</a> <a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">discarded messages information : notifyDiscardedMessages</a>
<b>Outputs</b>	<a href="#">response : notifyDiscardedMessagesResponse</a>
<b>Error Cases</b>	<a href="#">business exception : notifyDiscardedMessagesException</a>
<b>Activity Diagram</b>	<a href="#">BO-*::Handle discarded messages information</a>

## 5.2.2 Set service as available for a participant

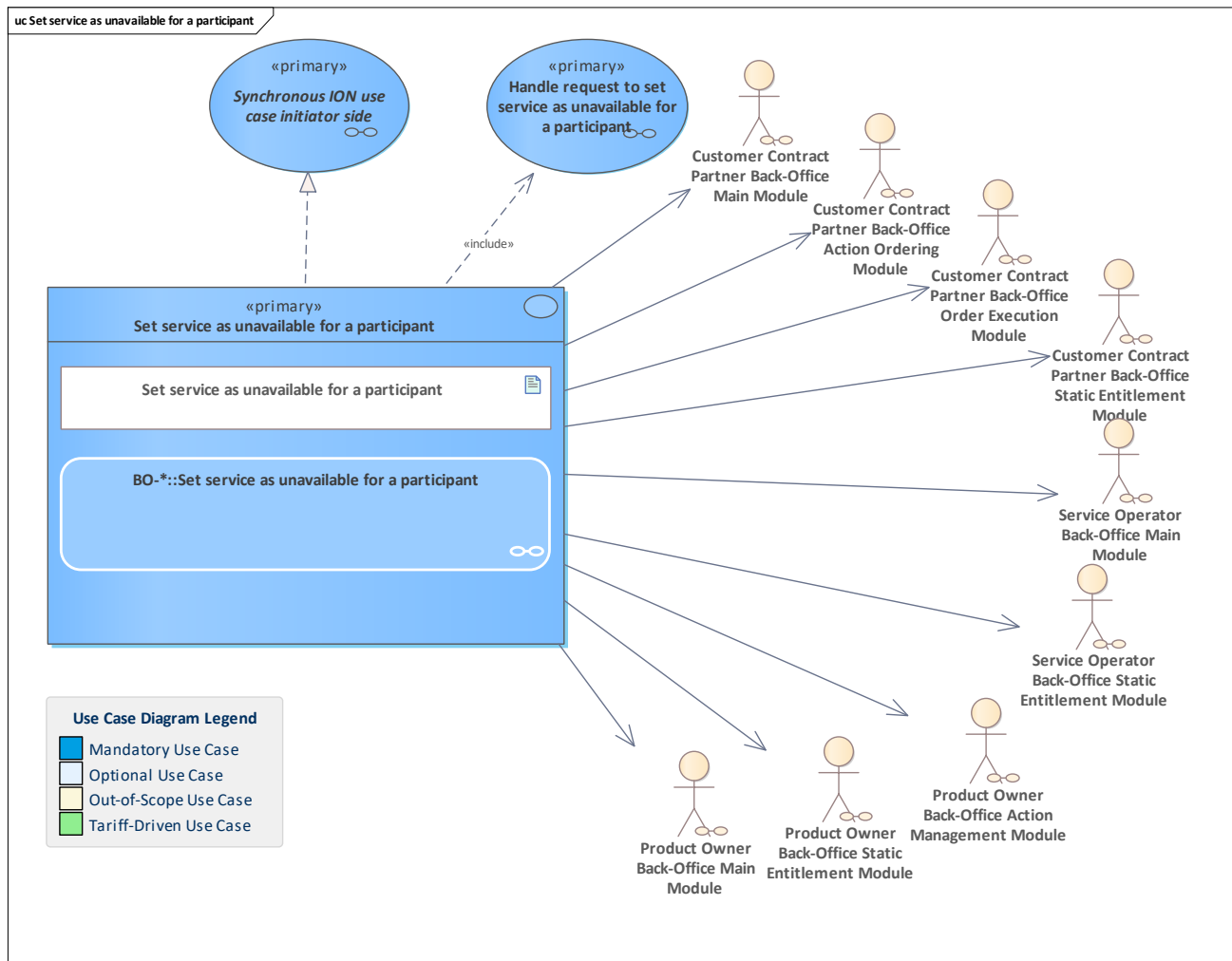


<b>Use Case</b>	<a href="#">Set service as available for a participant</a>
<b>Description</b>	In this use case, a participant sets a certain service as being available in the central routing engine (CRE). The purpose of this use case is to enable receiving messages for use cases in an asynchronous context. Note: if a service is announced to be available (again) the CRE will send stored messages to this service if any stored messages are still in the CRE's message queue.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Action Ordering Module</a> <a href="#">Customer Contract Partner Back-Office Order Execution Module</a> <a href="#">Customer Contract Partner Back-Office Static Entitlement Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Service Operator Back-Office Static Entitlement Module</a> <a href="#">Product Owner Back-Office Action Management Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Static Entitlement Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	



<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle request to set service as available for a participant</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-*::Set service as available for a participant</a>

## 5.2.3 Set service as unavailable for a participant

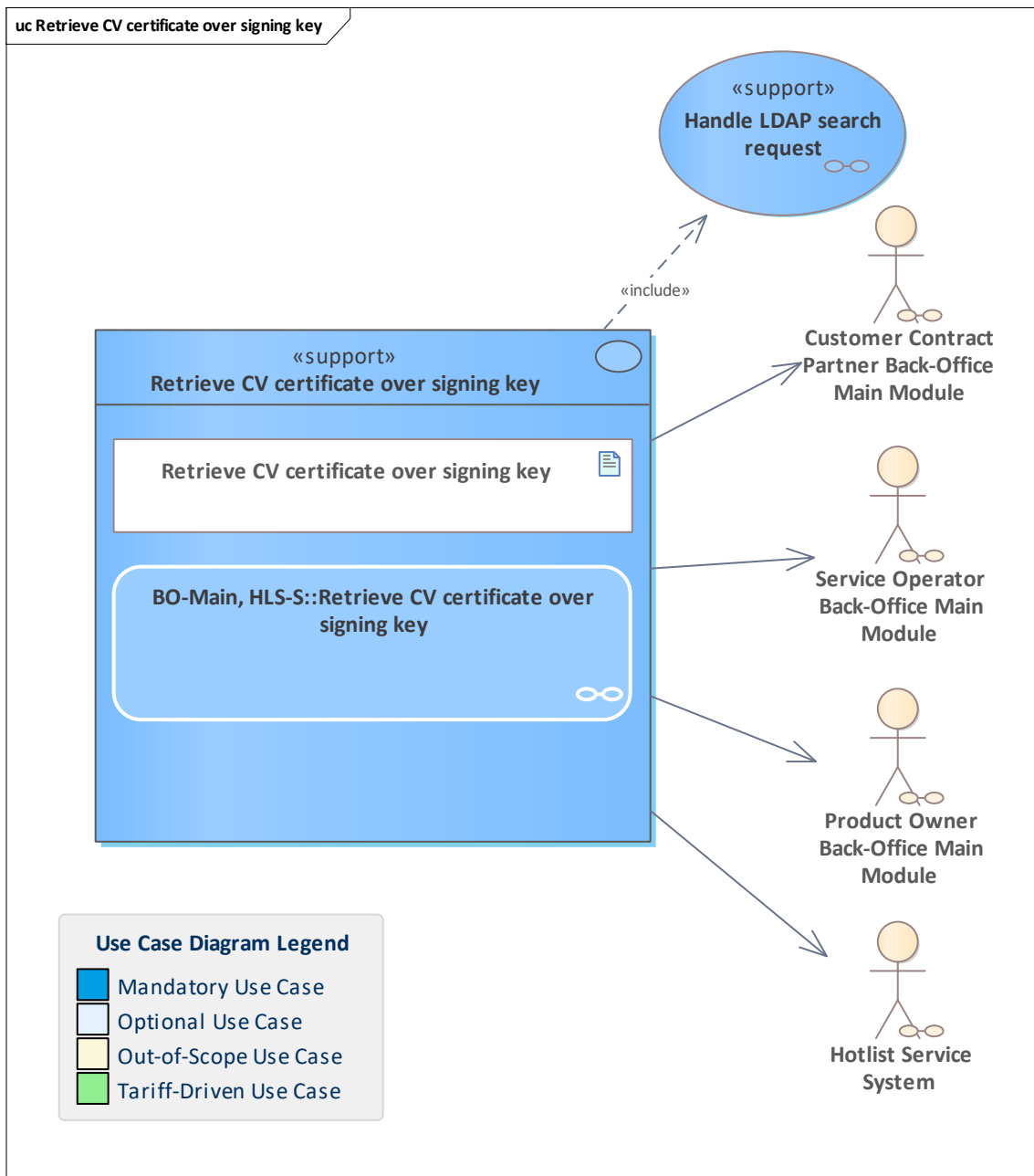


<b>Use Case</b>	<u>Set service as unavailable for a participant</u>
<b>Description</b>	<p>In this use case, a participant sets a certain service as being unavailable in the central routing engine (CRE).</p> <p>The purpose of this use case is to disable receiving messages for use cases in an asynchronous context. This means, that from now on, the CRE will store messages for a later delivery instead of routing them directly to the system that implements the service.</p> <p><b>Note:</b> participants must configure their services via the ESH in a preparatory step.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u> <u>Customer Contract Partner Back-Office Action Ordering Module</u> <u>Customer Contract Partner Back-Office Order Execution Module</u> <u>Customer Contract Partner Back-Office Static Entitlement Module</u> <u>Service Operator Back-Office Main Module</u> <u>Service Operator Back-Office Static Entitlement Module</u> <u>Product Owner Back-Office Static Entitlement Module</u> <u>Product Owner Back-Office Main Module</u> <u>Product Owner Back-Office Action Management Module</u>
<b>Preconditions</b>	



<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle request to set service as unavailable for a participant</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-*::Set service as unavailable for a participant</a>

## 5.2.4 Retrieve CV certificate over signing key



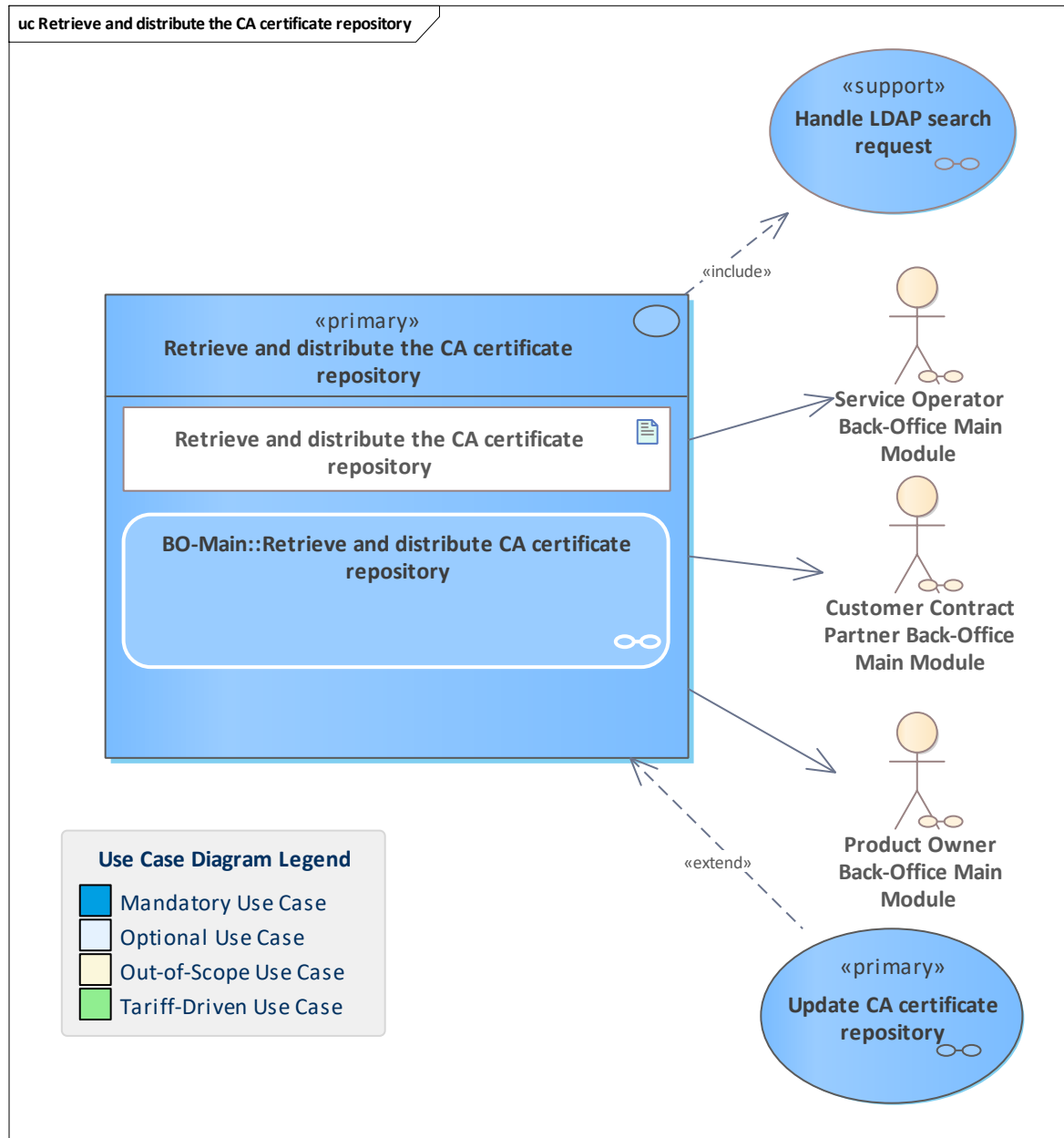
<b>Use Case</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Description</b>	Retrieve the latest certificate over the signing key of an end entity. Note that there might be several certificates for this end entity that are not relevant here: superseded certificates and certificates over keys not used for signature purposes.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Hotlist Service System</a>
<b>Preconditions</b>	





<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle LDAP search request</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">App instance ID : AppInstanceId</a>
<b>Outputs</b>	<a href="#">CV certificate over signing key : CvCertificate</a>
<b>Error Cases</b>	<a href="#">M-PKI not reachable</a> <a href="#">Unknown app instance ID</a>
<b>Activity Diagram</b>	<a href="#">BO-Main, HLS-S::Retrieve CV certificate over signing key</a>

## 5.2.5 Retrieve and distribute the CA certificate repository

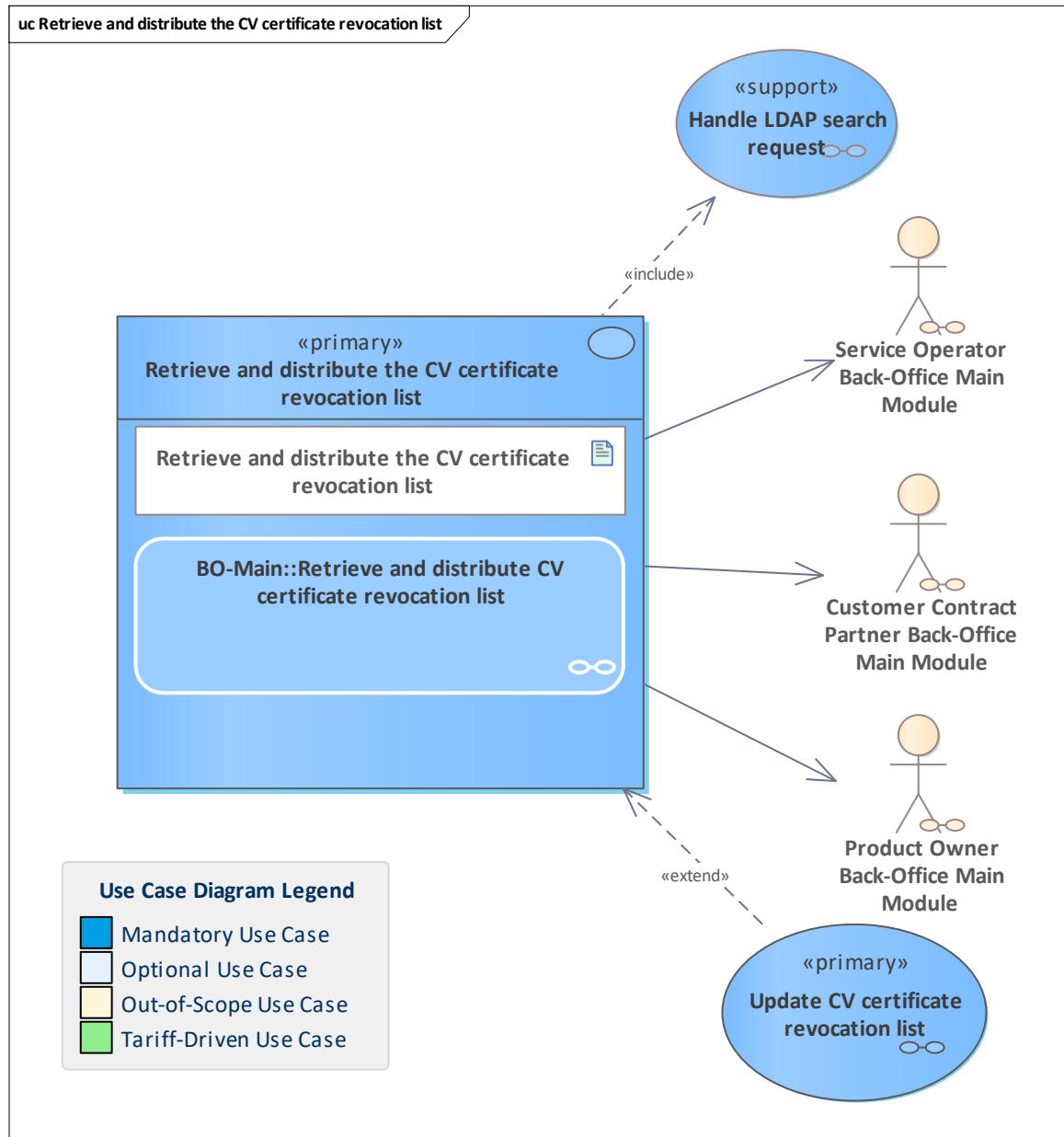


<b>Use Case</b>	<a href="#">Retrieve and distribute the CA certificate repository</a>
<b>Description</b>	The back-office system retrieves the CA certificate repository from the media-PKI (M-PKI). If the requestor operates terminals that belong to back-office this system, the CA certificate repository is also updated in all of them. This process needs to run periodically to keep the CA certificate repository up to date.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>



	Product Owner Back-Office Main Module
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Update CA certificate repository</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle LDAP search request</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Retrieve and distribute CA certificate repository</a>

## 5.2.6 Retrieve and distribute the CV certificate revocation list

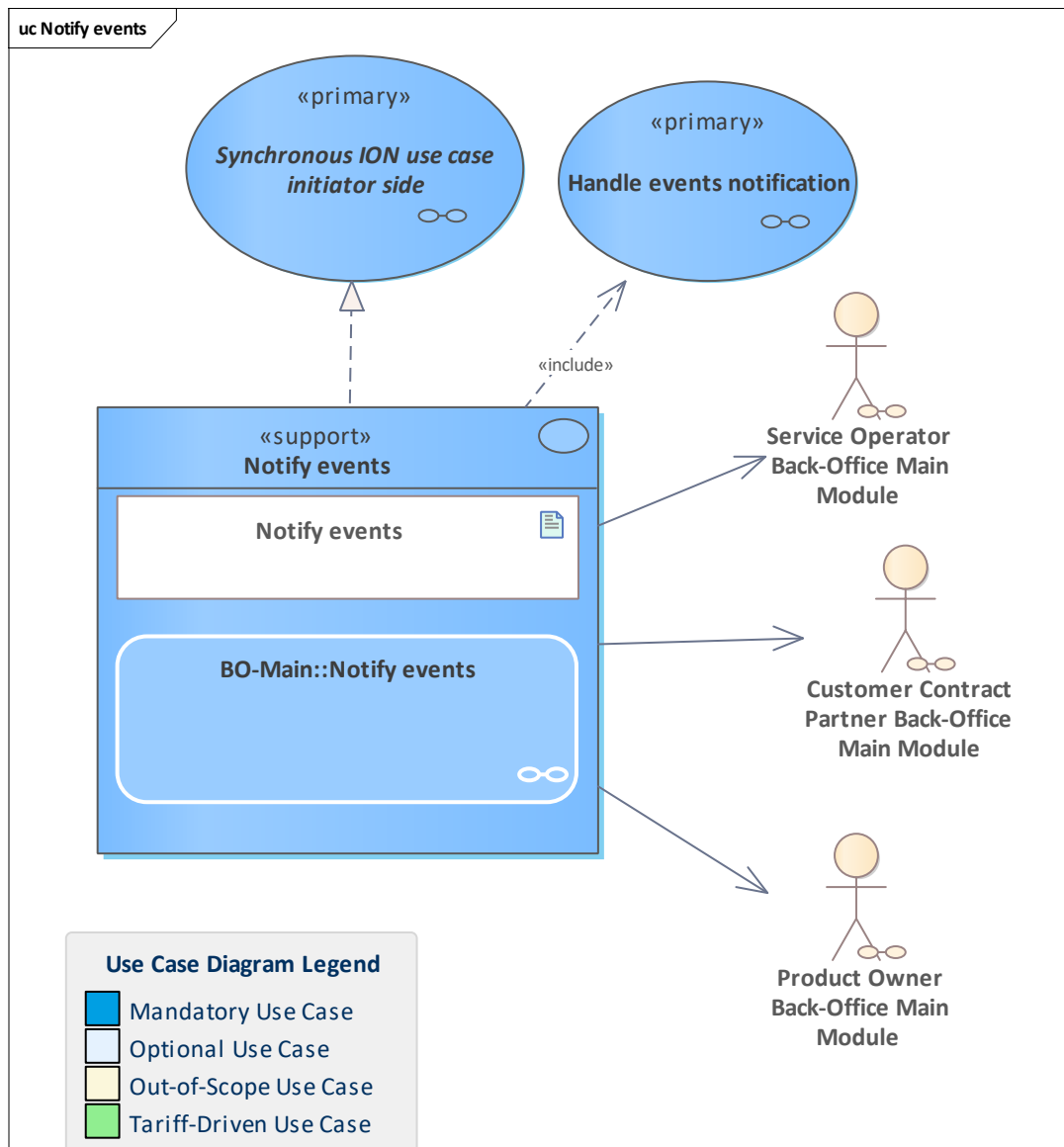


<b>Use Case</b>	<a href="#">Retrieve and distribute the CV certificate revocation list</a>
<b>Description</b>	The back-office system retrieves the CV certificate revocation list from the media-PKI (M-PKI). If the requestor operates terminals that belong to back-office this system, the CV certificate revocation list is also updated in all of them. This process needs to run periodically to keep the CV certificate revocation list up to date.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Product Owner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>



<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Update CV certificate revocation list</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle LDAP search request</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Retrieve and distribute CV certificate revocation list</a>

## 5.2.7 Notify events

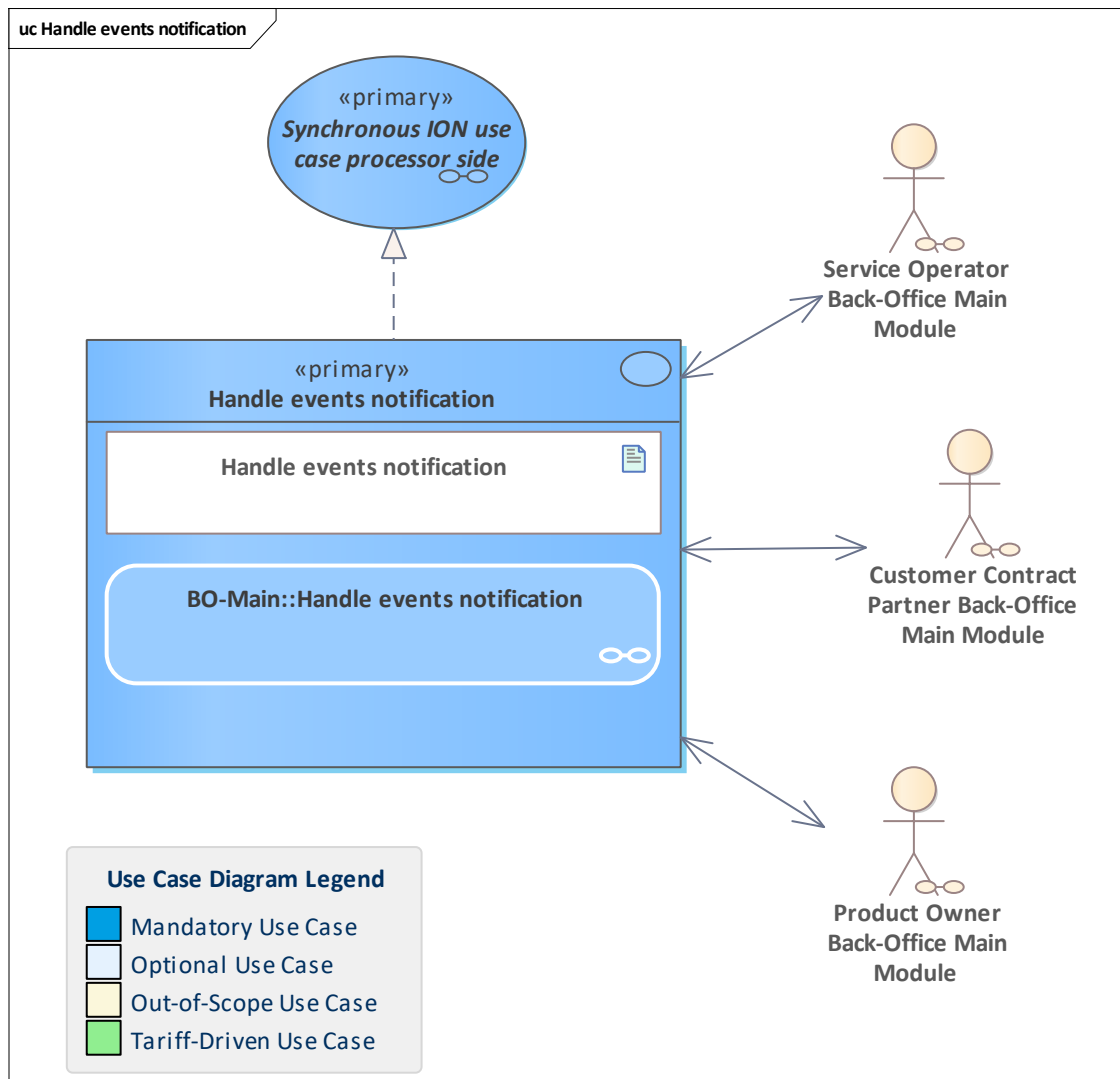


<b>Use Case</b>	<a href="#">Notify events</a>
<b>Description</b>	Notify a participant about warnings which occurred during the downstream monitoring or similar.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle events notification</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>



<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Role : PartnerRoleCode</a> <a href="#">Warnings : Warning</a> <a href="#">Receiver : OrganisationId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Notify events</a>

## 5.2.8 Handle events notification



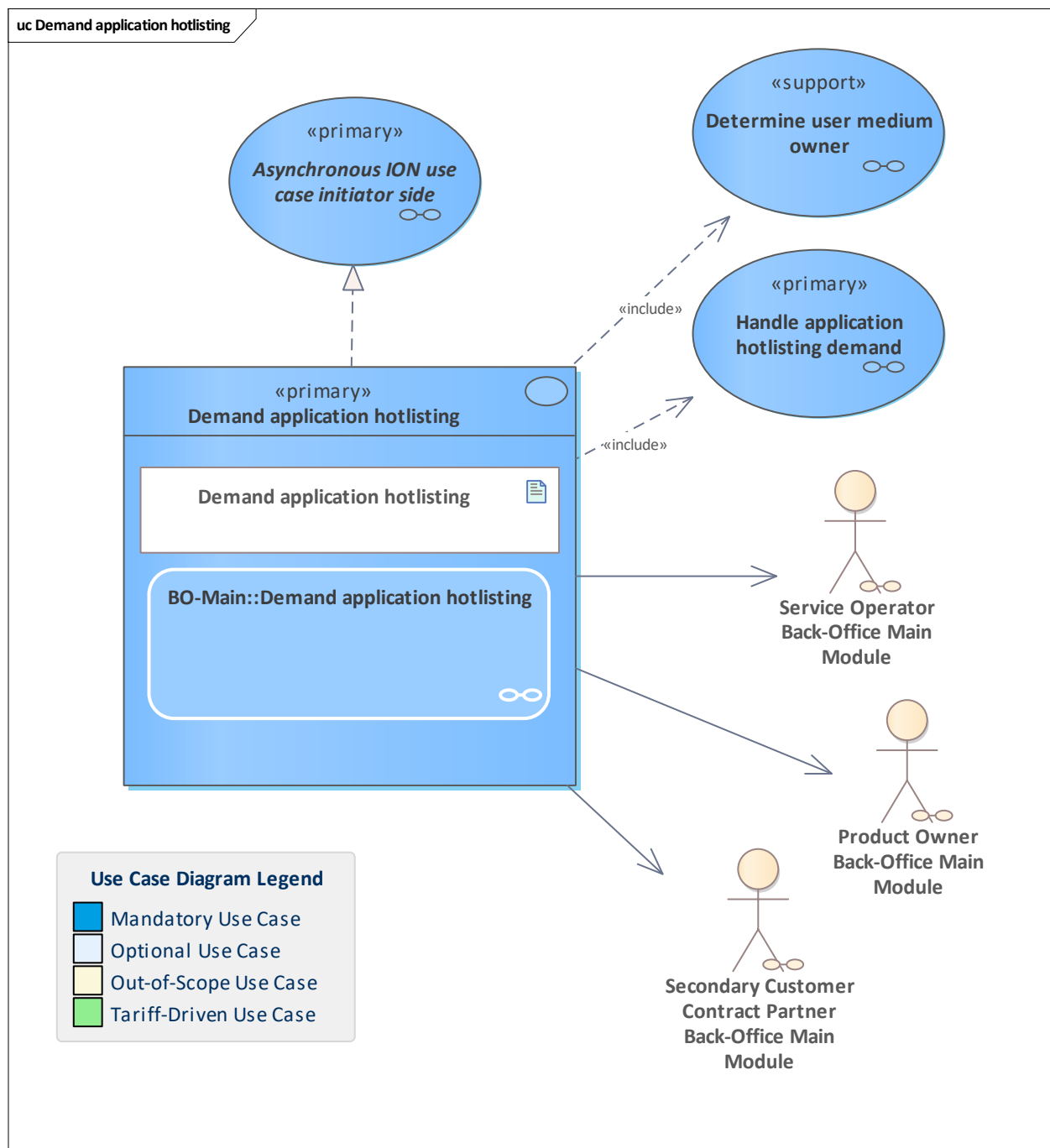
<b>Use Case</b>	<a href="#">Handle events notification</a>
<b>Description</b>	A participant is informed about warnings.
<b>Initiating Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case processor side</a>
<b>Base Activity</b>	





<b>Inputs</b>	<a href="#">Notify events : notifyEvents</a>
<b>Outputs</b>	<a href="#">Notify events response : notifyEventsResponse</a>
<b>Error Cases</b>	<a href="#">Notify events exception : notifyEventsException</a>
<b>Activity Diagram</b>	<a href="#">BO-Main::Handle events notification</a>

## 5.2.9 Demand application hotlisting

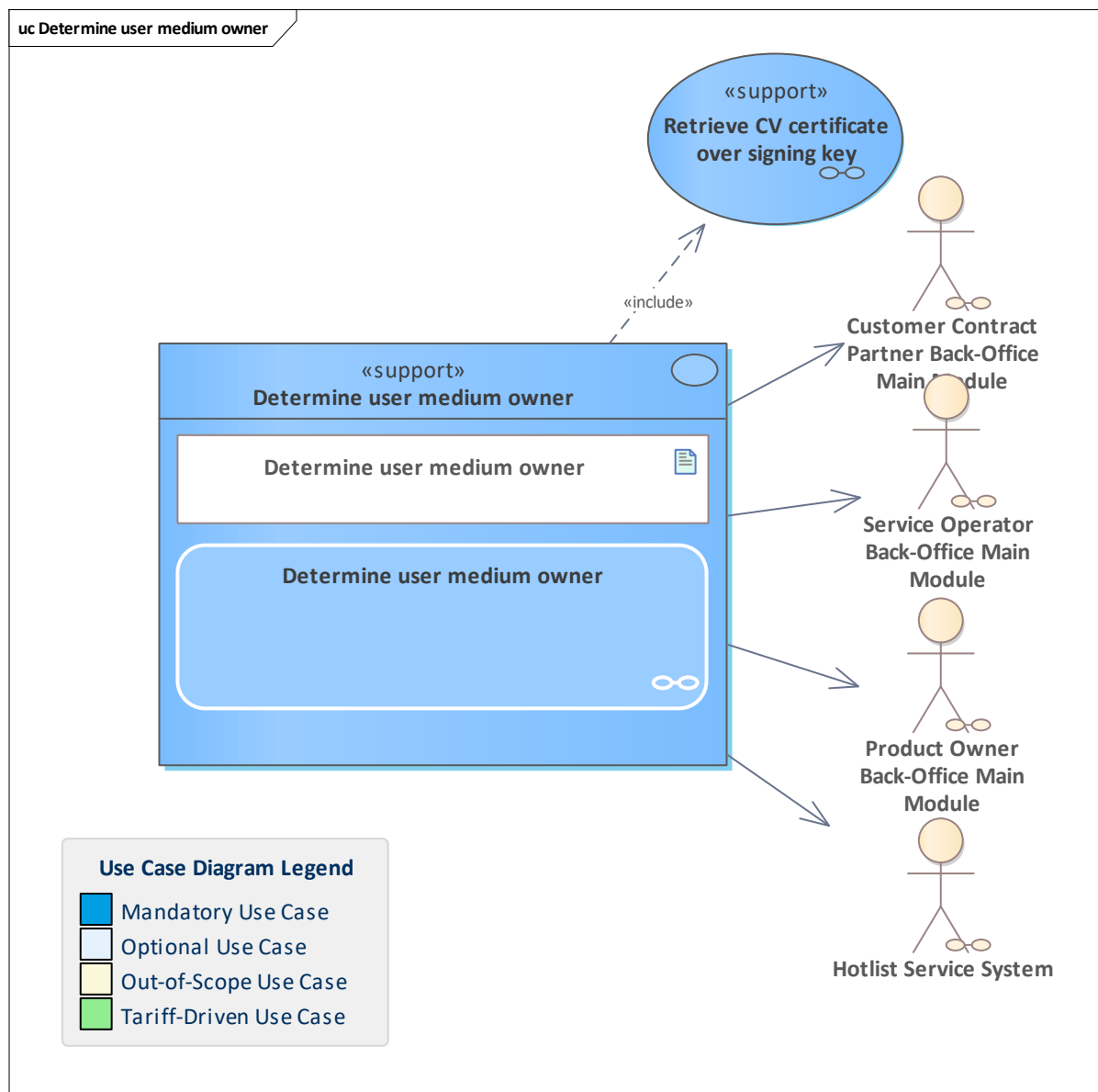


Use Case	Demand application hotlisting
Description	<p>The PO, sCCP or SO as sender demands the pCCP that issued the application to the customer to hotlist the application in question. The sender adds a reason (SO e.g. in case of a defective user medium, sCCP e.g. in case of a lost user medium) and further hotlisting parameters.</p> <p>This application can be either a user medium application with an application instance ID or a MOTICS app with an SCE ID.</p>



	In most cases, the hotlist demand from a third party will be caused by monitoring.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle application hotlisting demand</a> / <a href="#">Determine user medium owner</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Application transition counter : TransitionCounter</a> <a href="#">Blocking reason : BlockingReason</a> <a href="#">Hotlist entry expiration time : ZonedDateTime</a> <a href="#">Application blocking mode : ApplicationBlockingModeCode</a> <a href="#">App instance ID : AppInstanceId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Demand application hotlisting</a>

## 5.2.10 Determine user medium owner

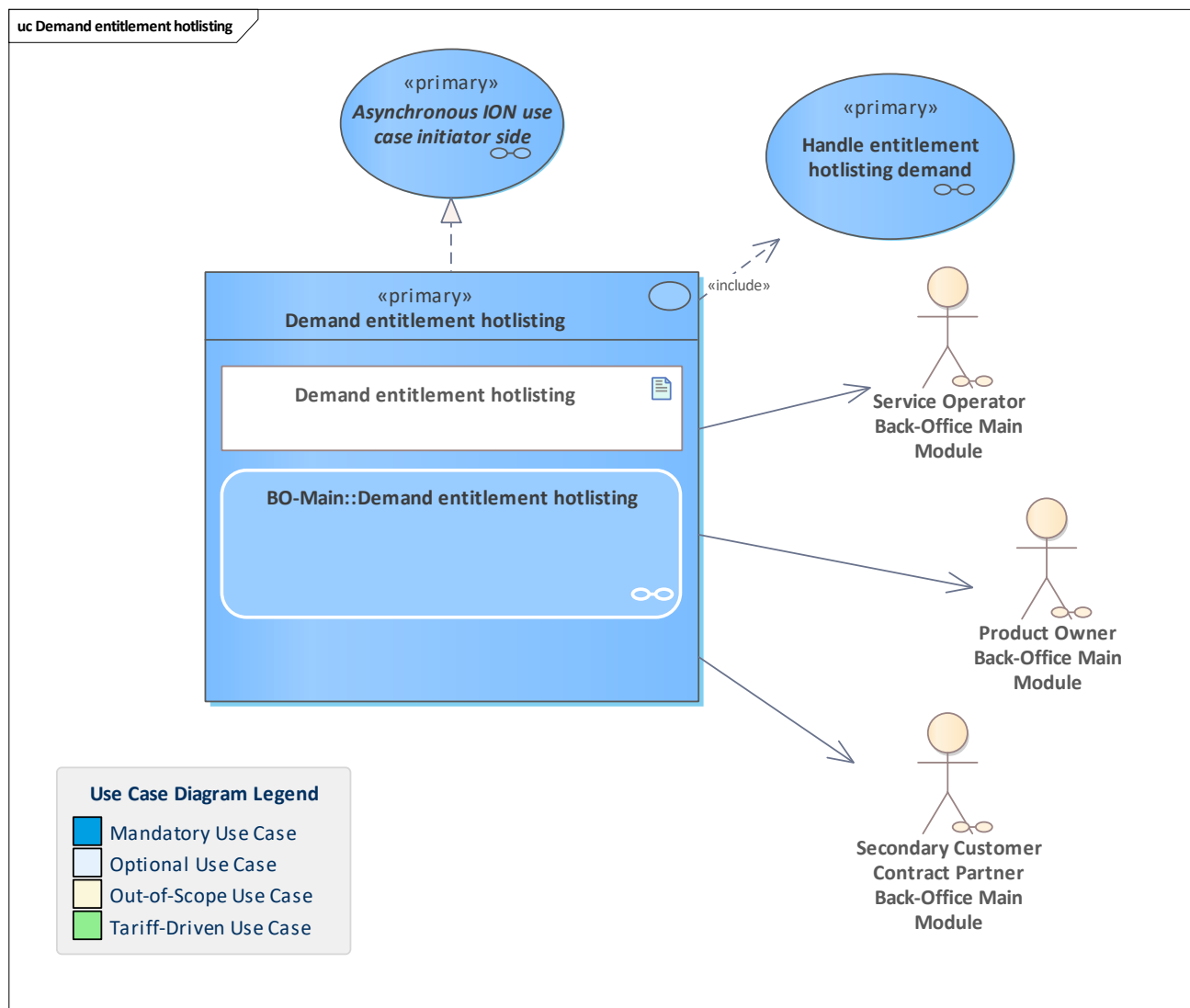


<b>Use Case</b>	<u>Determine user medium owner</u>
<b>Description</b>	<p>Determine the owner of a user medium with an application or SCE (in case of MOTICS with static entitlement) in a back-office system using the ownership information contained in the corresponding certificate.</p> <p>To determine the user medium owner, the matching CV certificate is to be fetched, so the caller retrieves the latest certificate over the signing key of an end entity.</p> <p>Note that there might be several certificates for this end entity, that are not relevant here: superseded certificates and certificates for keys not used for signature purposes. Thus, the right certificate that delivers the owner organisation ID has to be filtered.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u>



	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Hotlist Service System</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">App instance ID : AppInstanceId</a>
<b>Outputs</b>	<a href="#">Org ID of user medium owner : OrganisationId</a>
<b>Error Cases</b>	<a href="#">M-PKI not reachable</a> <a href="#">Unknown app instance ID</a>
<b>Activity Diagram</b>	<a href="#">Determine user medium owner</a>

## 5.2.11 Demand entitlement hotlisting

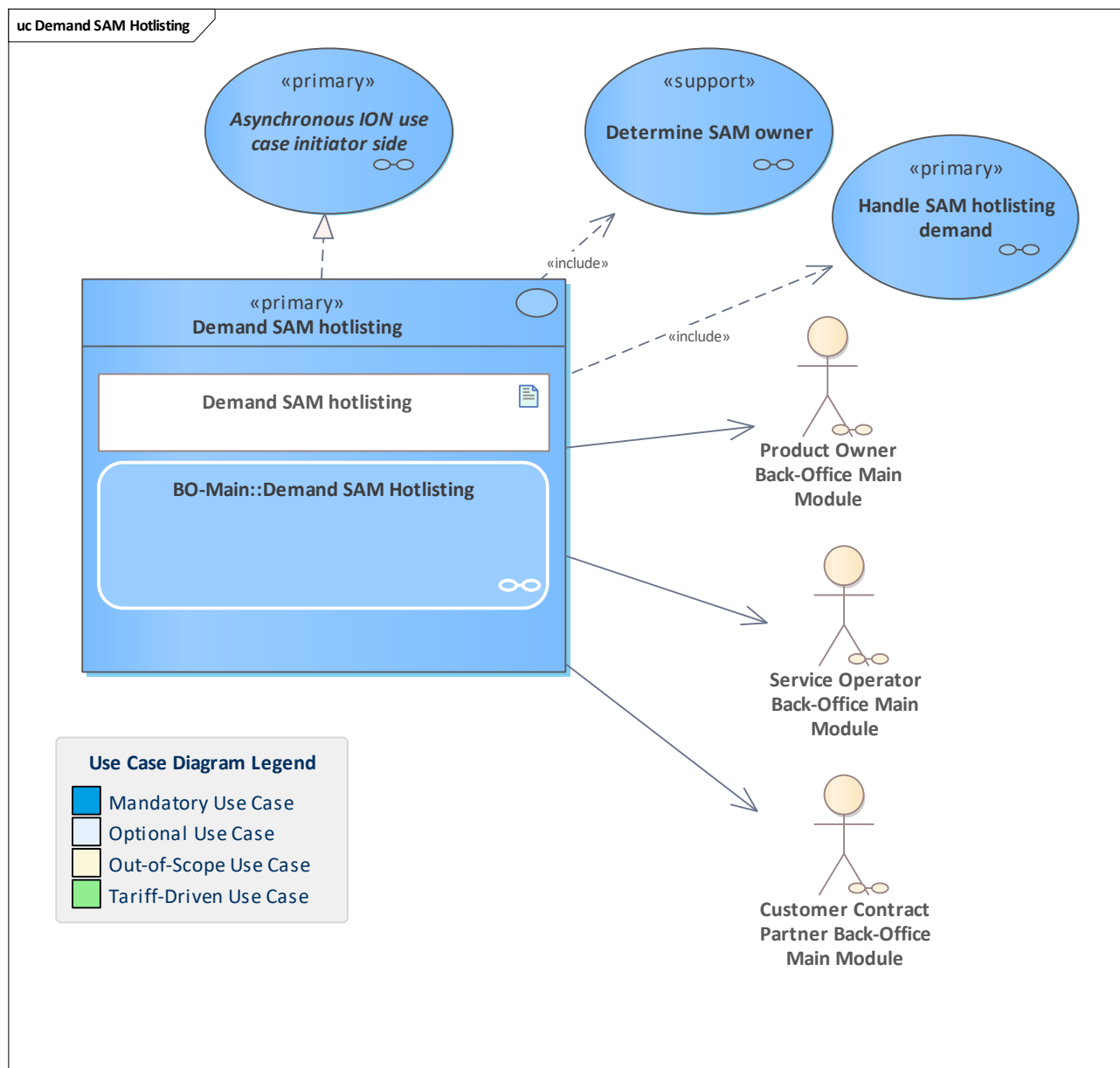


Use Case	Demand entitlement hotlisting
Description	<p>In this use case, the PO, sCCP or SO as sender demands the pCCP that issued the entitlement to the customer to hotlist the entitlement in question.</p> <p>The sender adds a reason and further hotlisting parameters. This entitlement can be either located on a user medium with an application or a static entitlement coming from a barcode or a MOTICS app.</p> <p>In most cases, the hotlisting demand from a third party will be caused by monitoring.</p> <p>Possible reasons are:</p> <ul style="list-style-type: none"> <li>• Inconsistencies have occurred during monitoring (the most likely reason)</li> <li>• Offence against terms of carriage</li> <li>• Payment delay</li> <li>• Referenced product was deactivated</li> </ul>



	<ul style="list-style-type: none"><li>Entitlement with the same ID already exists</li><li>User medium/application which contains entitlements of a secondary customer contract partner (CCP) was replaced</li></ul>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement hotlisting demand</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Blocking reason : BlockingReason</a> <a href="#">Hotlist entry expiration time : ZonedDateTime</a> <a href="#">Entitlement blocking mode : EntitlementBlockingModeCode</a> <a href="#">Hotlist entry effective time : ZonedDateTime</a> <a href="#">UM or SCE-ID : AppInstanceId</a> <a href="#">Entitlement transition counter : TransitionCounter</a> <a href="#">Product ID : ProductId</a> <a href="#">Entitlement ID : EntitlementId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Demand entitlement hotlisting</a>

## 5.2.12 Demand SAM hotlisting



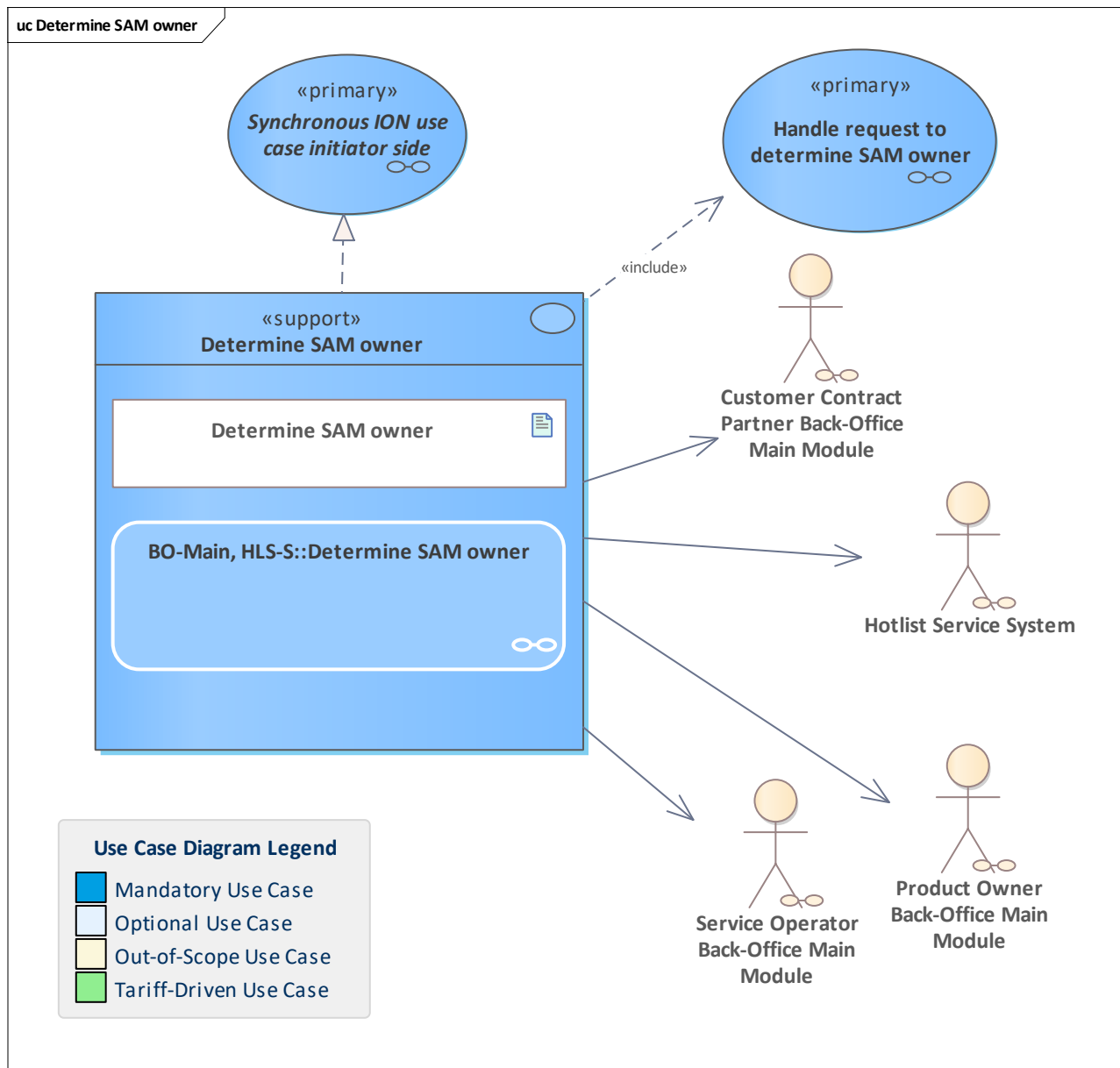
<b>Use Case</b>	<a href="#">Demand SAM hotlisting</a>
<b>Description</b>	<p>A PO, SO or CCP sends a demand for hotlisting to the SAM Owner (SO or CCP). Reasons for this demand could either be loss or theft of a SAM that was used in one of the SO or CCP terminals. Another reason could be suspected fraud, which could be detected by monitoring.</p> <p>Before demanding the SAM hotlisting, the demander must find out the SAM owner to place the demand to the right receiver.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	





<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle SAM hotlisting demand</a> / <a href="#">Determine SAM owner</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Blocking reason : BlockingReason</a> <a href="#">SAM action counter : ActionCounter</a> <a href="#">SAM ID : AppInstanceId</a> <a href="#">SAM entitlement issuance counter : EntitlementIssuanceCounter</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Demand SAM Hotlisting</a>

## 5.2.13 Determine SAM owner

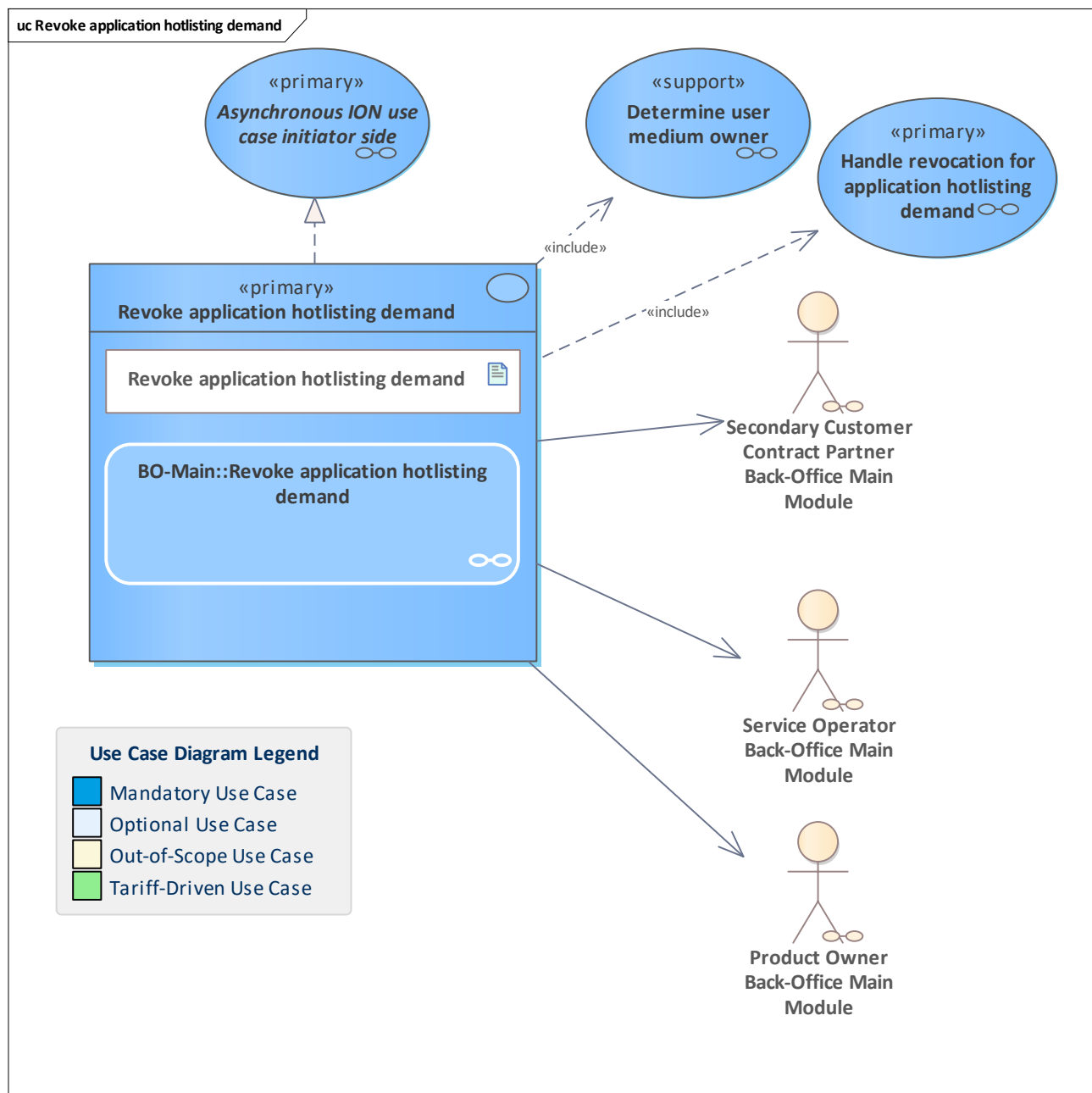


<b>Use Case</b>	<a href="#">Determine SAM owner</a>
<b>Description</b>	Determine the SAM Owner (organisation ID and role) for a given SAM ID using the service provided by the ESH.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Hotlist Service System</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases</b>	<a href="#">Handle request to determine SAM owner</a>



<b>(Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">SAM ID : AppInstanceId</a>
<b>Outputs</b>	<a href="#">SAM owner : OrganisationalUnit</a>
<b>Error Cases</b>	<a href="#">E_CO_APP_INSTANCE_ID_UNKNOWN</a> <a href="#">Unknown SAM : E_CO_APP_INSTANCE_ID_UNKNOWN</a> <a href="#">Timeout</a>
<b>Activity Diagram</b>	<a href="#">BO-Main, HLS-S::Determine SAM owner</a>

## 5.2.14 Revoke application hotlisting demand

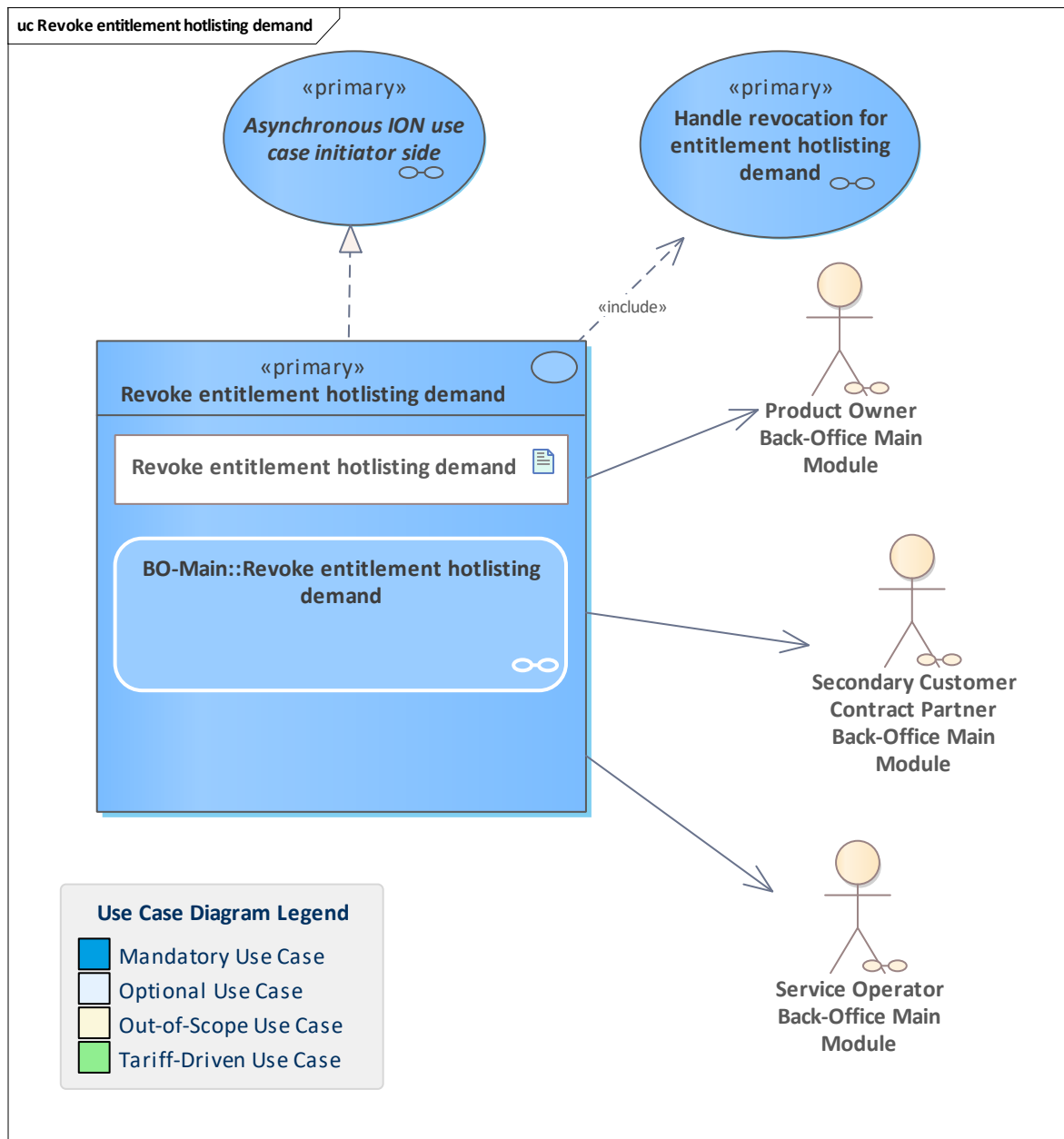


Use Case	Revoke application hotlisting demand
Description	<p>An SO, PO or sCCP sends a demand for an application hotlisting revocation to the pCCP since the blocking reason no longer exists. The pCCP will check the revocation demand and, in case of a positive decision, have the application removed from the hotlist. The revocation references the ION message of the previous hotlisting demand. This is a rarely used use case.</p> <p><b>Note:</b> if the application owner has changed, the new application will be determined by <a href="#">Determine user medium owner</a>.</p>



	The new application owner must have transferred the information about old hotlisting demands (if any), especially the ION message references.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle revocation for application hotlisting demand</a> / <a href="#">Determine user medium owner</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">ION message id : IonMessageId</a> <a href="#">App instance ID : AppInstanceId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Revoke application hotlisting demand</a>

## 5.2.15 Revoke entitlement hotlisting demand

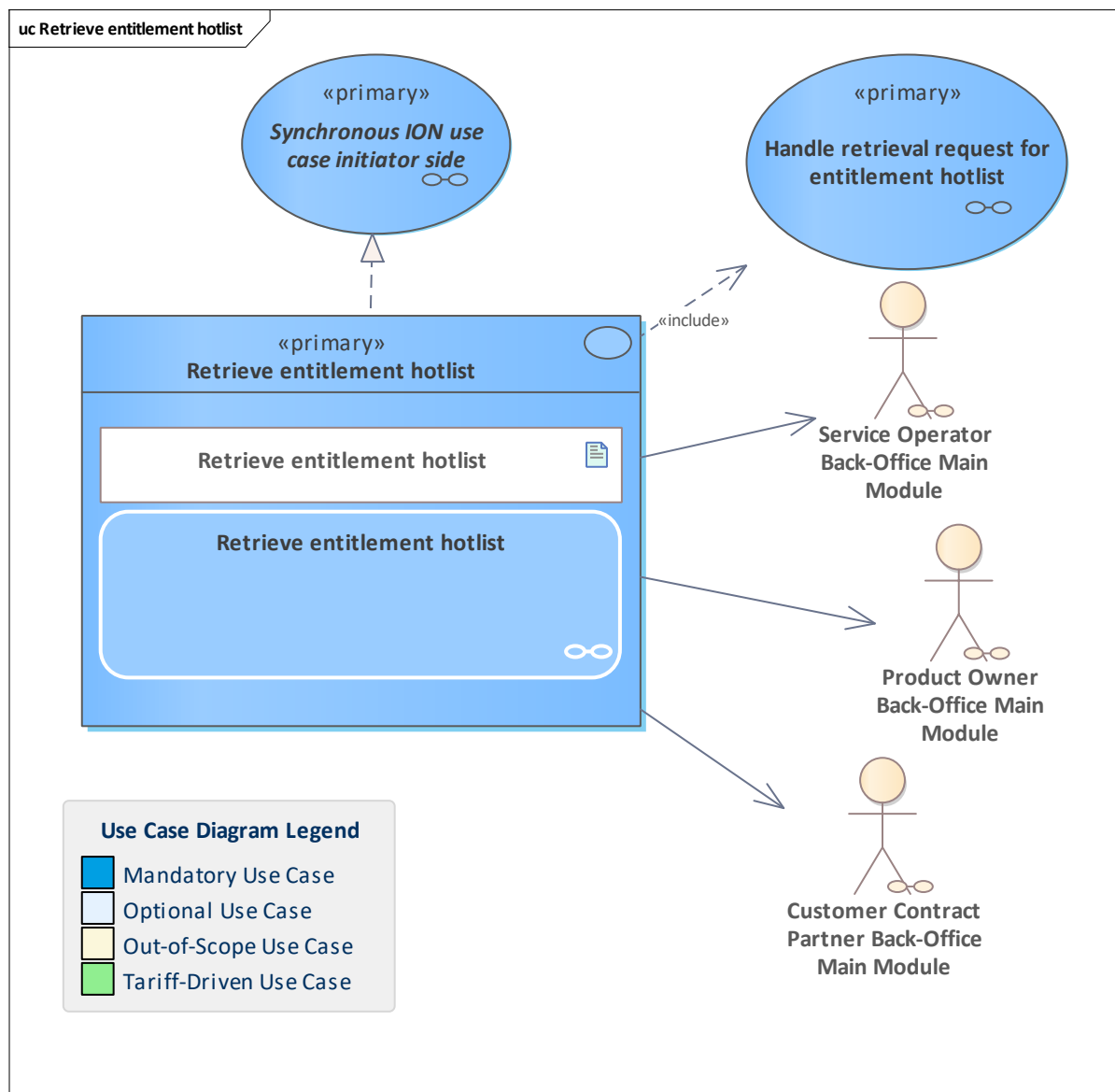


<b>Use Case</b>	<a href="#">Revoke entitlement hotlisting demand</a>
<b>Description</b>	<p>A PO, SO or sCCP sends a demand for hotlisting revocation to the entitlement owner (pCCP).</p> <p>The pCCP will check the revocation demand and, in case of a positive decision, have the entitlement removed from the hotlist. The revocation references the ION message of the previous hotlisting demand.</p> <p>This is a rarely used use case.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>



<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle revocation for entitlement hotlisting demand</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Entitlement Id : EntitlementId</a> <a href="#">ION message ID : IonMessageId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Revoke entitlement hotlisting demand</a>

## 5.2.16 Retrieve entitlement hotlist



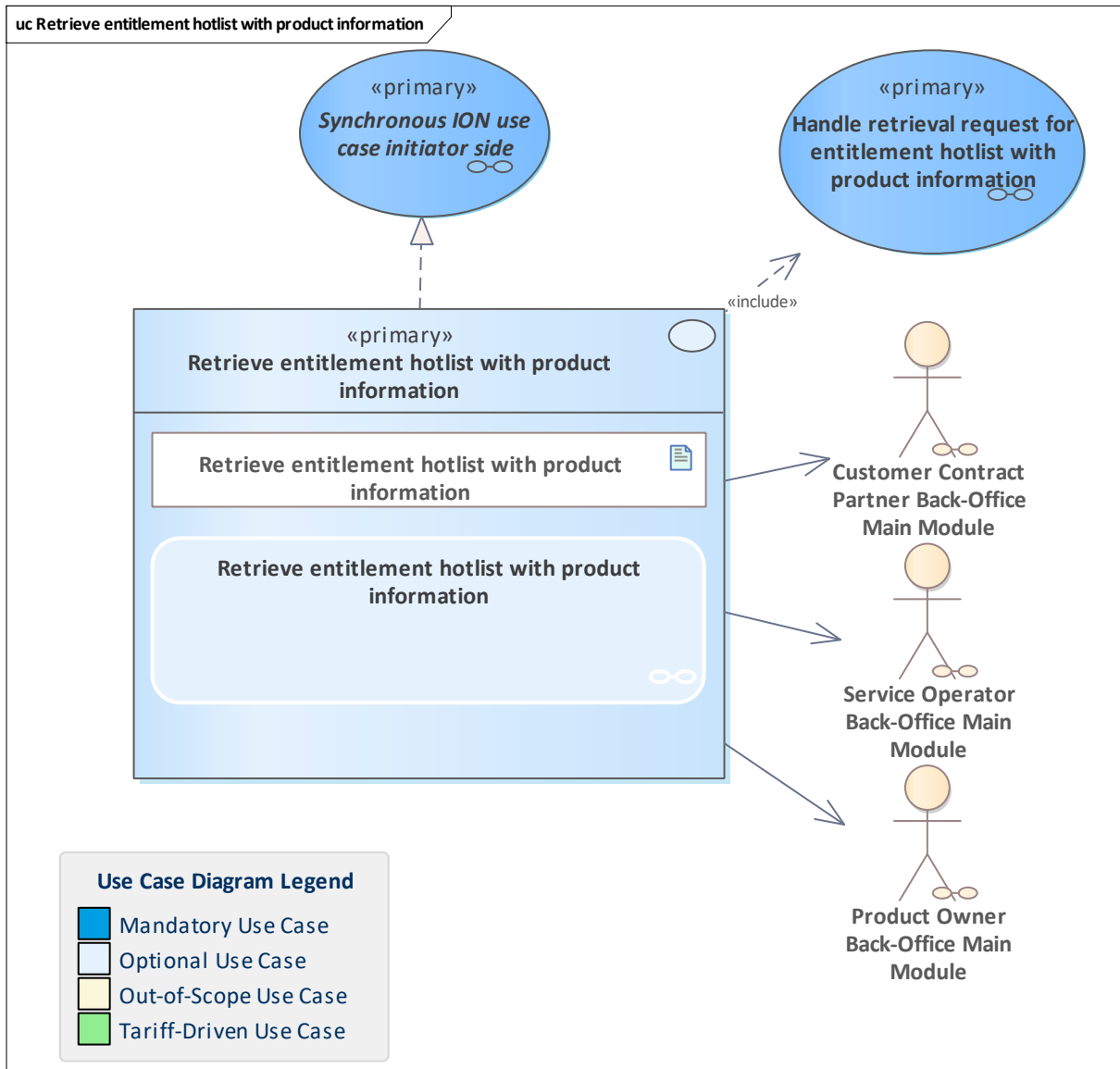
<b>Use Case</b>	<u>Retrieve entitlement hotlist</u>
<b>Description</b>	The SO, PO and CCP want to update their entitlement hotlist inventory by retrieving the current entitlement hotlist from the hotlist service system. Please note that the entitlement hotlist can be retrieved either as an incremental or a total hotlist, as well as a total hotlist with product information.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u> <u>Service Operator Back-Office Main Module</u> <u>Product Owner Back-Office Main Module</u>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	





<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for entitlement hotlist</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a>
<b>Outputs</b>	<a href="#">Entitlement hotlist : EntitlementHotlist</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Retrieve entitlement hotlist</a>

## 5.2.17 Optional: Retrieve entitlement hotlist with product information

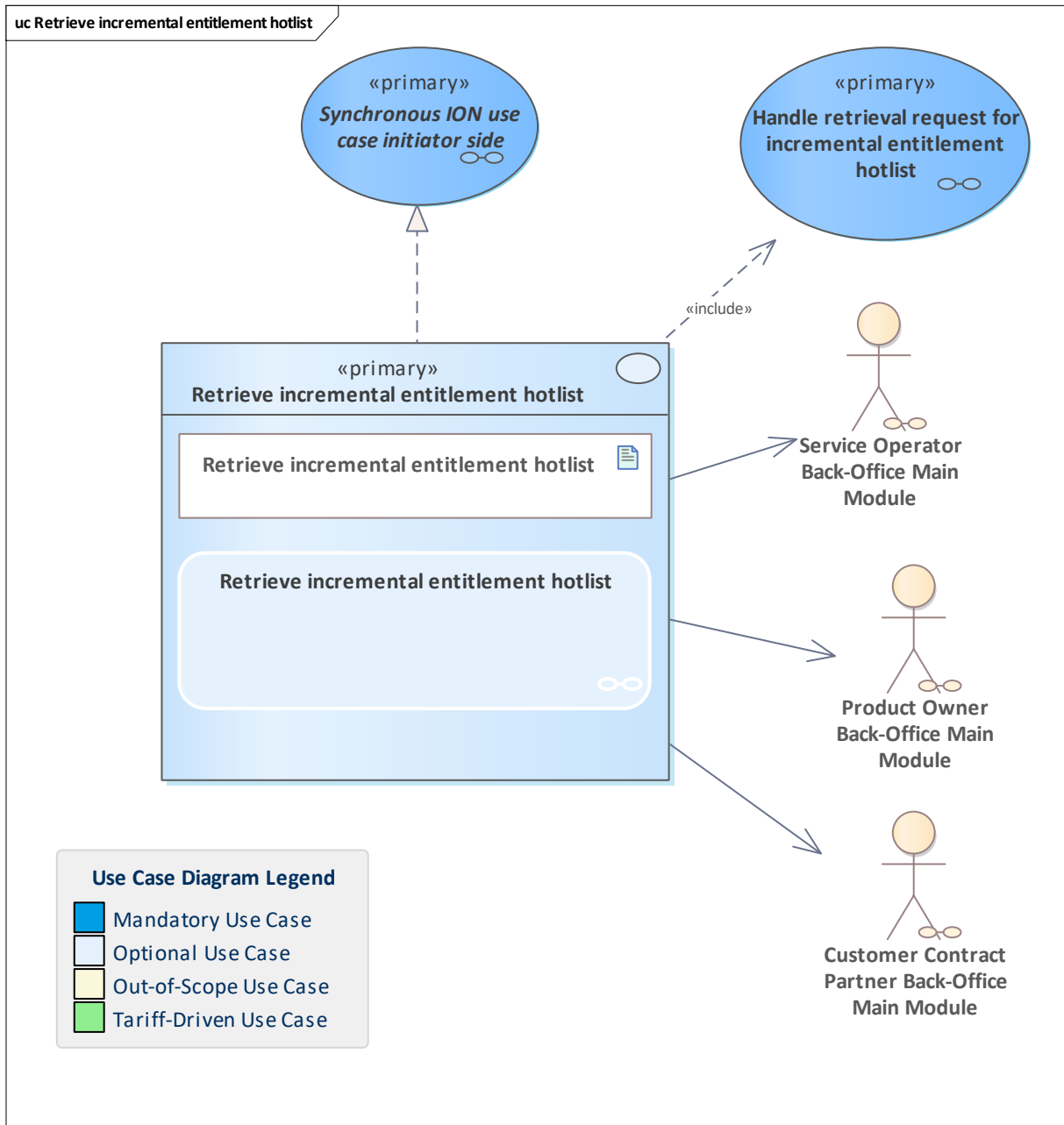


<b>Use Case</b>	<a href="#">Retrieve entitlement hotlist with product information</a>
<b>Description</b>	The SO, PO and CCP want to update their entitlement hotlist inventory by retrieving the entitlement hotlist with additionally contained product information from the hotlist service system. Please note that the entitlement hotlist can be retrieved either as an incremental or a total hotlist, as well as a hotlist with product information (this use case).
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	



<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for entitlement hotlist with product information</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a>
<b>Outputs</b>	<a href="#">Entitlement hotlist : EntitlementHotlist</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Retrieve entitlement hotlist with product information</a>

## 5.2.18 Optional: Retrieve incremental entitlement hotlist

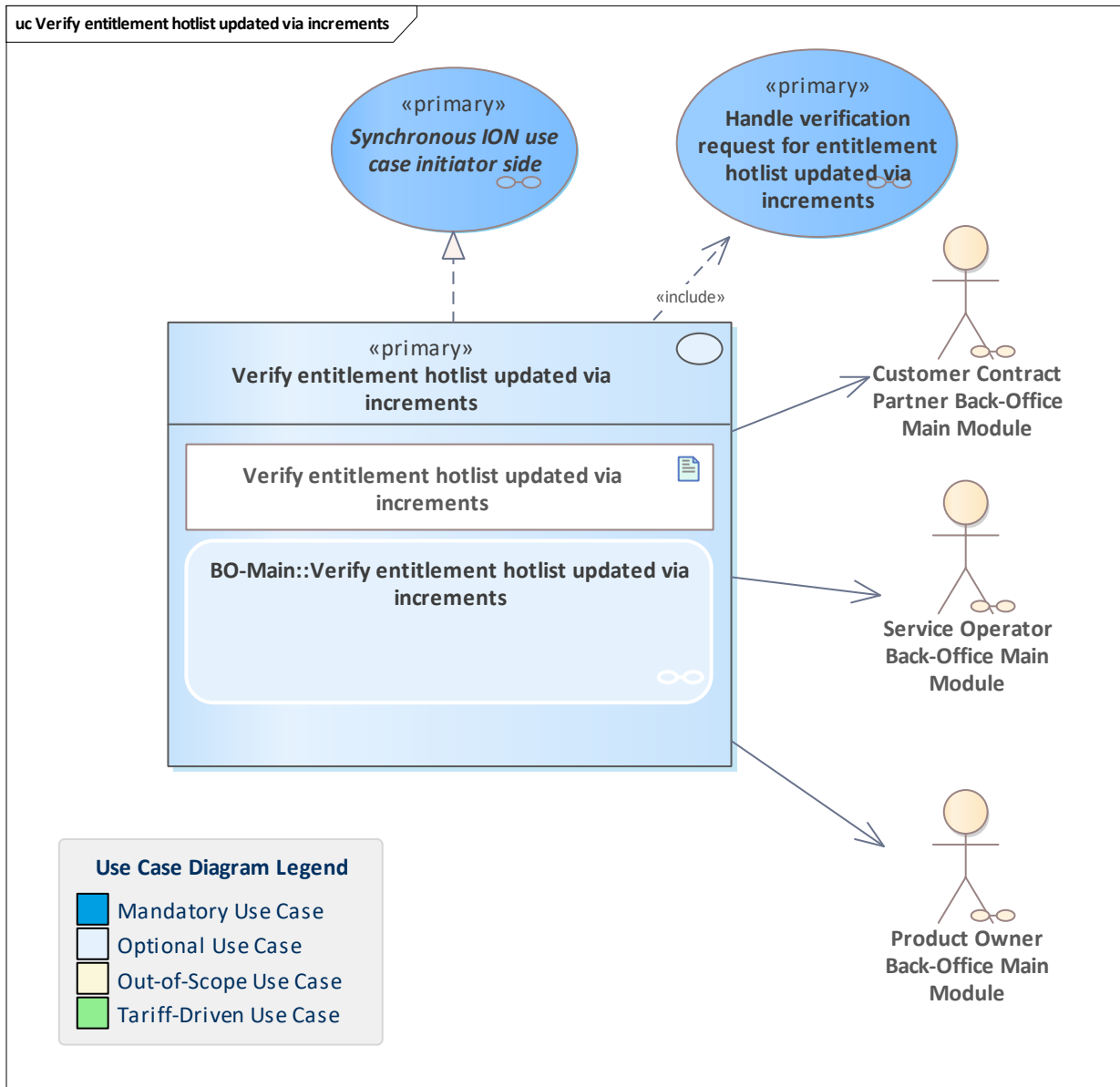


<b>Use Case</b>	<a href="#">Retrieve incremental entitlement hotlist</a>
<b>Description</b>	The SO, PO and CCP want to update their entitlement hotlist inventory by retrieving the current incremental entitlement hotlist from the hotlist service system. Please note that the entitlement hotlist can be retrieved either as an incremental (this use case) or a total hotlist, as well as a hotlist with product information.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a>



	<a href="#">Product Owner Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for incremental entitlement hotlist</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a>
<b>Outputs</b>	<a href="#">Updated entitlement hotlist : EntitlementHotlist</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Retrieve incremental entitlement hotlist</a>

## 5.2.19 Optional: Verify entitlement hotlist updated via increments

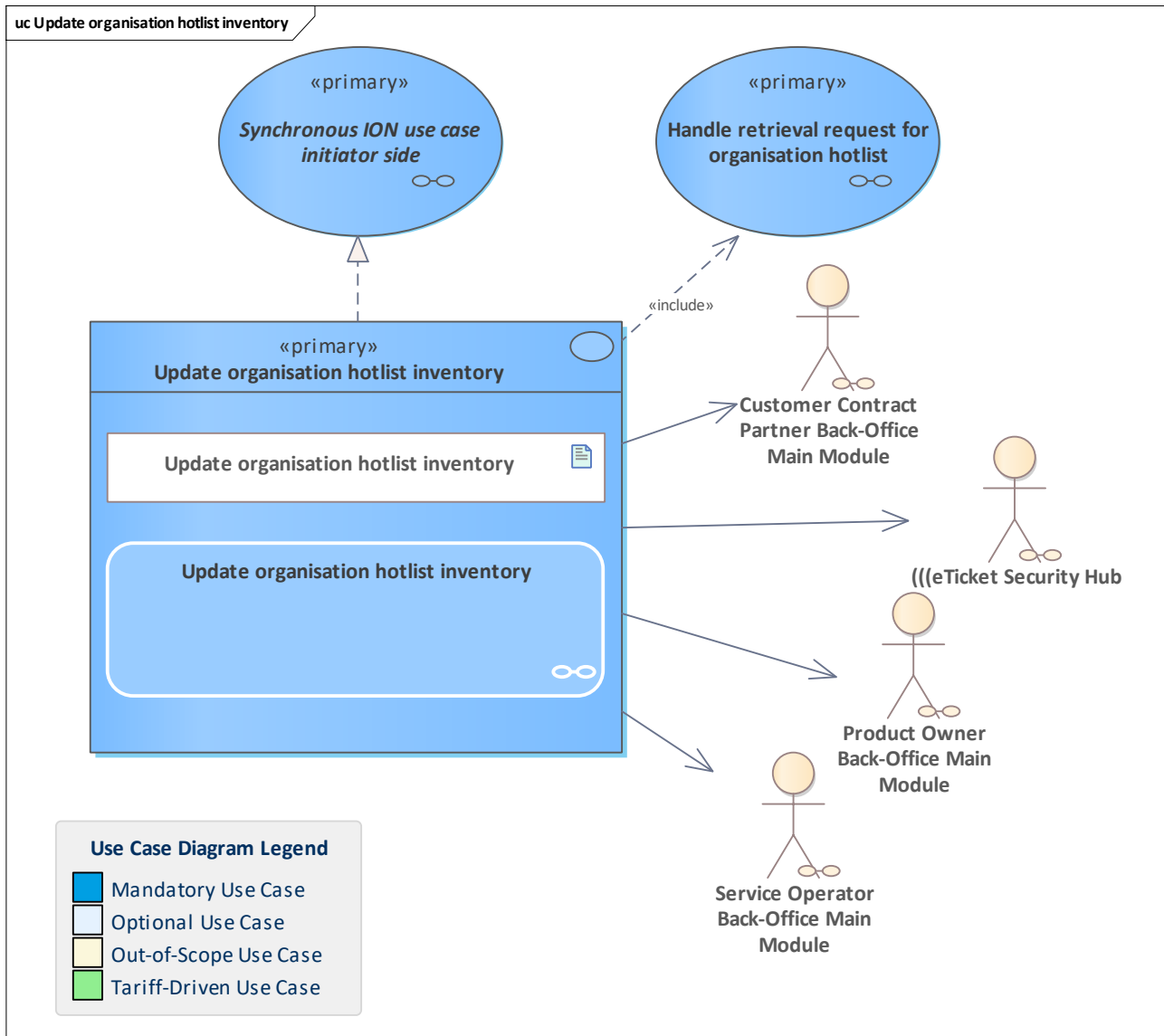


<b>Use Case</b>	<a href="#">Verify entitlement hotlist updated via increments</a>
<b>Description</b>	After updating the entitlement hotlist inventory via the last incremental entitlement hotlist, the participant must ensure that the updated hotlist inventory is the same as the hotlist inventory of the hotlist service system (filtered to the participant's products). For that reason, participants compute the checksum of all the entitlement IDs in their inventory and send it to the hotlist service system to verify the value of the checksum. See also <a href="#">Checksum calculation for hotlist and action list verification</a> and <a href="#">Example calculation for an entitlement hotlist inventory</a> .
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>



	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle verification request for entitlement hotlist updated via increments</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a> <a href="#">Updated entitlement hotlist : EntitlementHotlist</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Verify entitlement hotlist updated via increments</a>

## 5.2.20 Update organisation hotlist inventory



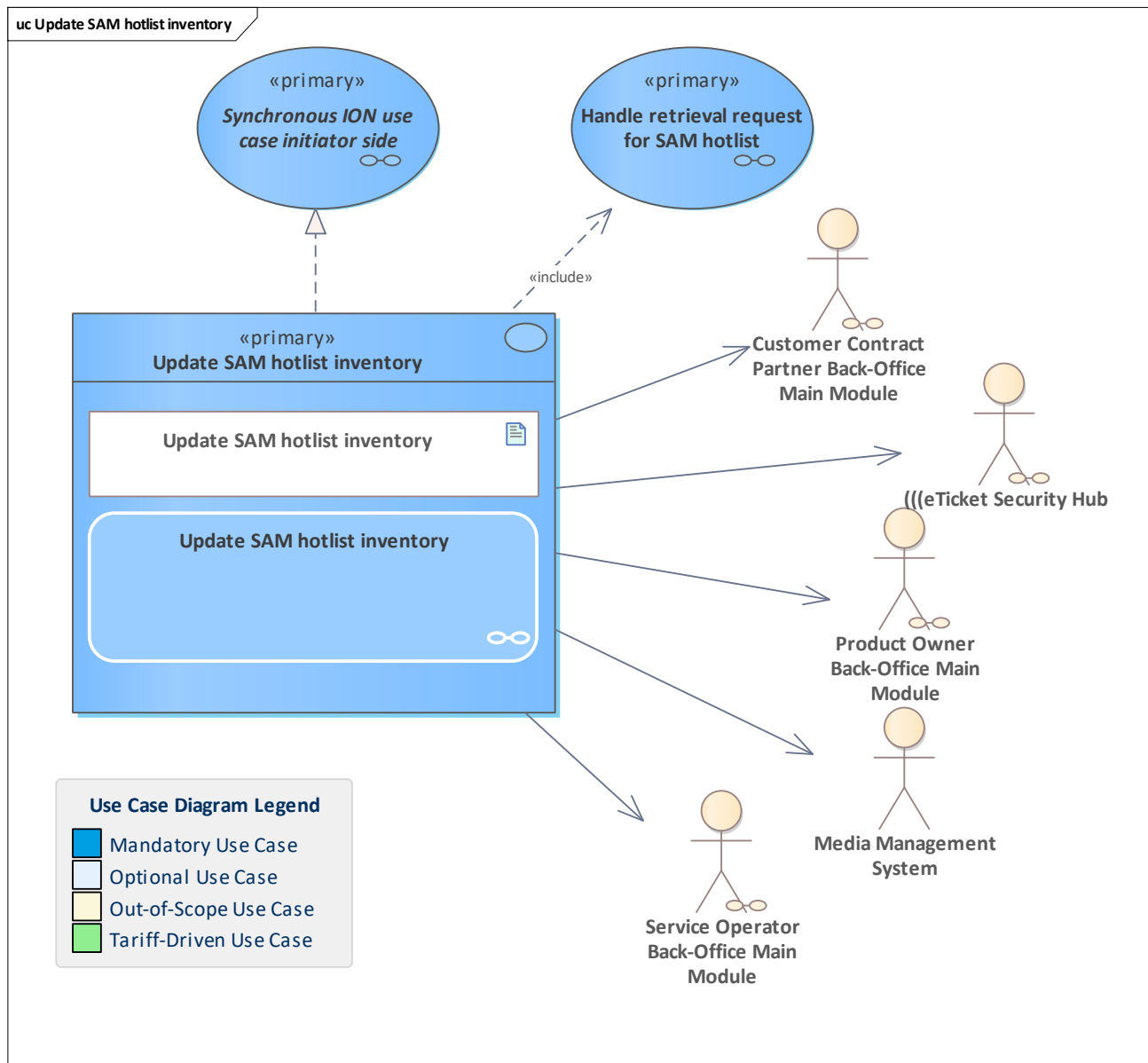
<b>Use Case</b>	<a href="#">Update organisation hotlist inventory</a>
<b>Description</b>	The SO, CCP, PO and the scheme manager's ESH want to update their organisation hotlist inventory by retrieving the current organisation hotlist from the hotlist service system and processing it into their organisation hotlist inventory.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">((eTicket Security Hub</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases</b>	<a href="#">Handle retrieval request for organisation hotlist</a>





<b>(Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	<a href="#">Updated organisation hotlist inventory</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Update organisation hotlist inventory</a>

## 5.2.21 Update SAM hotlist inventory

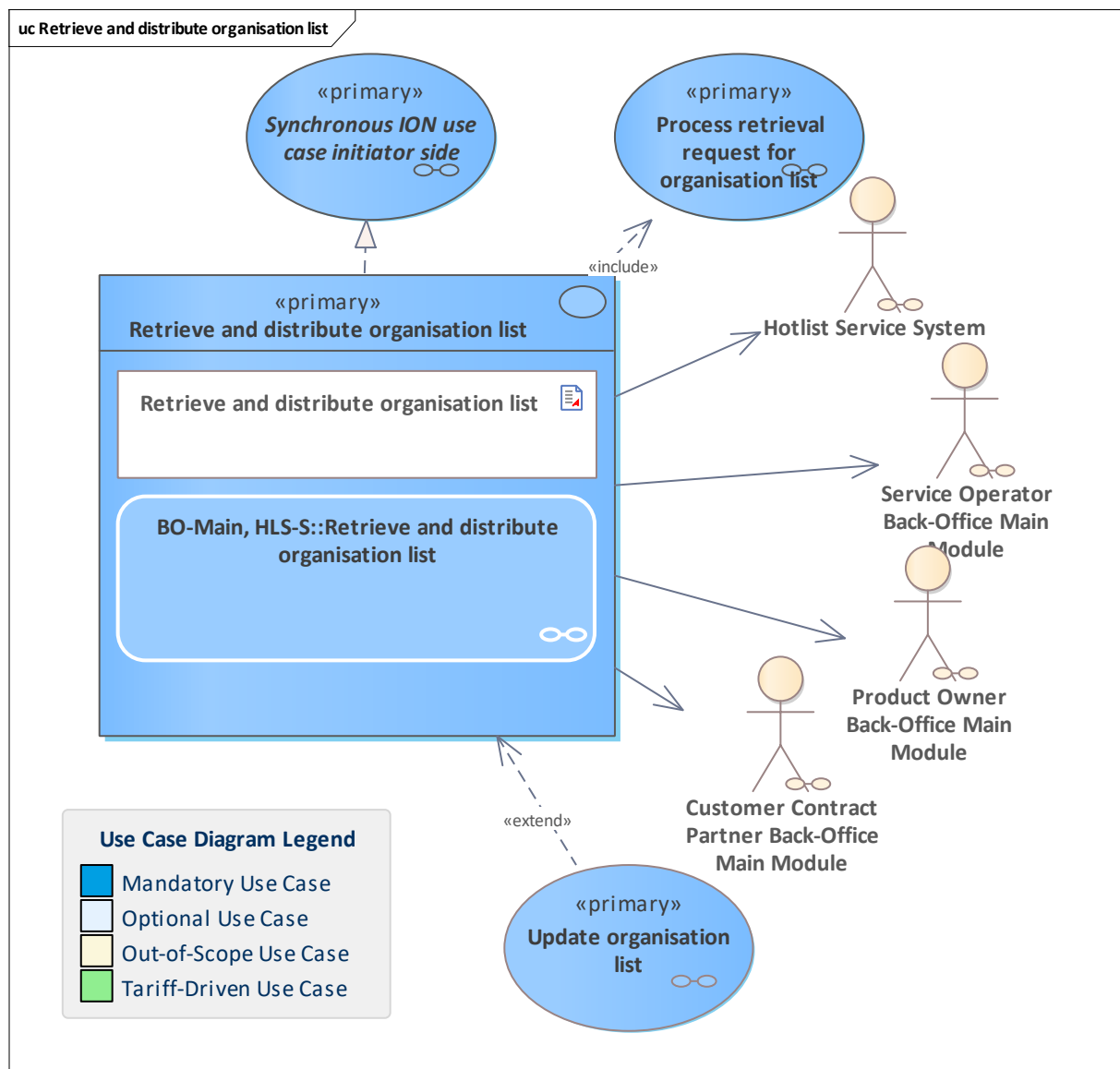


<b>Use Case</b>	<u>Update SAM hotlist inventory</u>
<b>Description</b>	The SO, CCP, PO and the scheme manager's ESH and MMS want to update their SAM hotlist inventory by retrieving the current SAM hotlist from the hotlist service system and processing it into the SAM hotlist inventory.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u> <u>Service Operator Back-Office Main Module</u> <u>Product Owner Back-Office Main Module</u> <u>Media Management System</u> <u>(((eTicket Security Hub</u>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	



<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for SAM hotlist</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	<a href="#">Updated SAM hotlist inventory</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Update SAM hotlist inventory</a>

## 5.2.22 Retrieve and distribute organisation list



<b>Use Case</b>	<a href="#">Retrieve and distribute organisation list</a>
<b>Description</b>	The registrar, as part of the scheme manager, provides a list of organisations. This list can be retrieved by all participants daily. An organisation list is distributed to the terminals by CCP and SO. Please note that the list does not involve any IP addresses of the organisations due to data protection.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Hotlist Service System</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	<a href="#">Update organisation list</a>



<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Process retrieval request for organisation list</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main, HLS-S::Retrieve and distribute organisation list</a>



## 6 Basic Bundle Terminal Operator System - Foundation

This functionality bundle contains use cases that all back-office systems of CCP and SO must implement. CCP and SO are terminal operators whose systems interact with the respective terminals.

### 6.1 Overview

Handle SAM hotlisting demand

Check and add SAM to hotlist

Retrieve application hotlist

Optional: Retrieve incremental application hotlist

Optional: Verify application hotlist updated via increments

Update authentication key hotlist inventory

Update hotlist inventory from operational perspective

Distribute SAM configuration

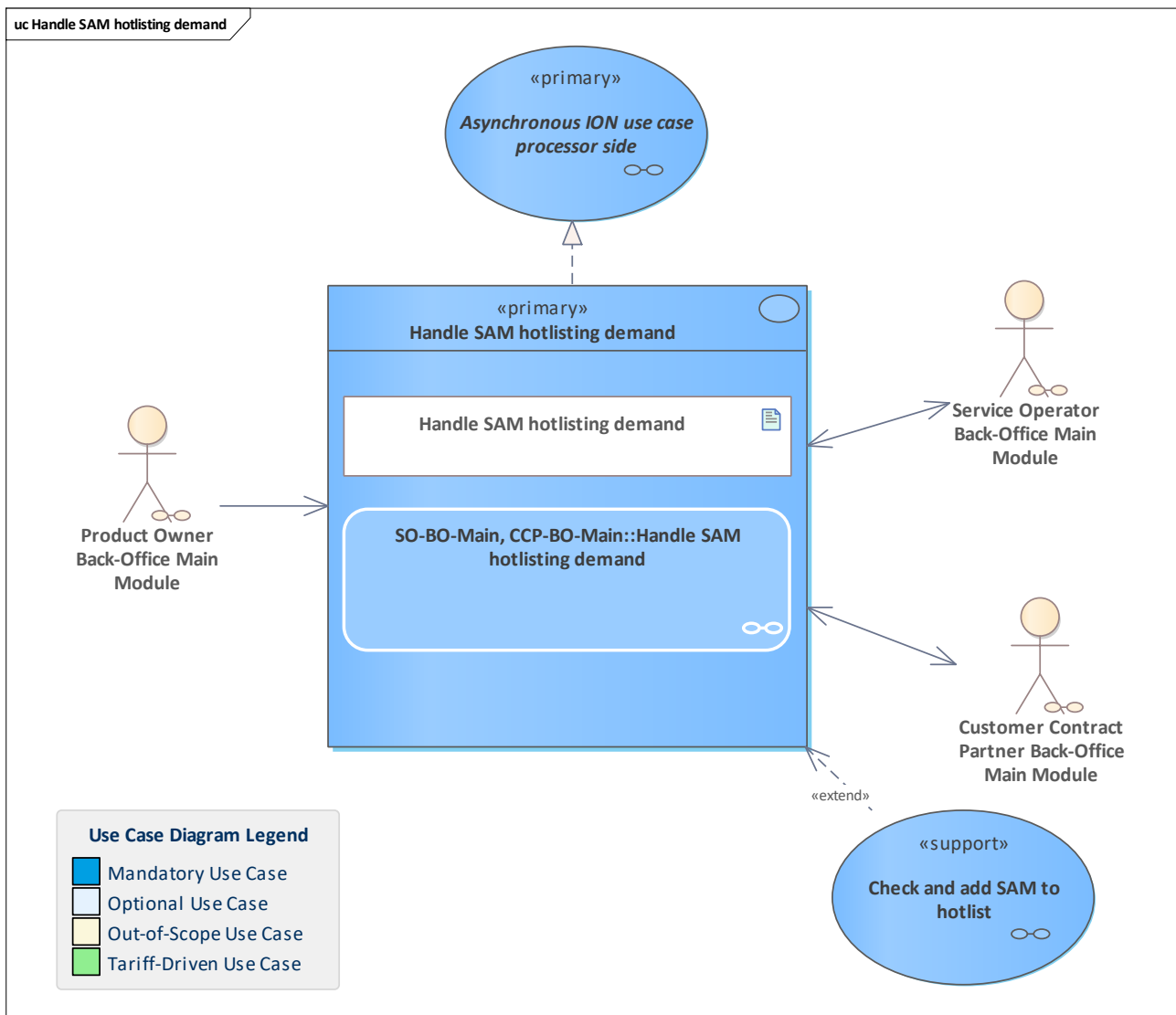
Optional: Distribute the SAM reset script

Distribute tariff module

Monitor SAMs from operational perspective

### 6.2 Use Cases

#### 6.2.1 Handle SAM hotlisting demand



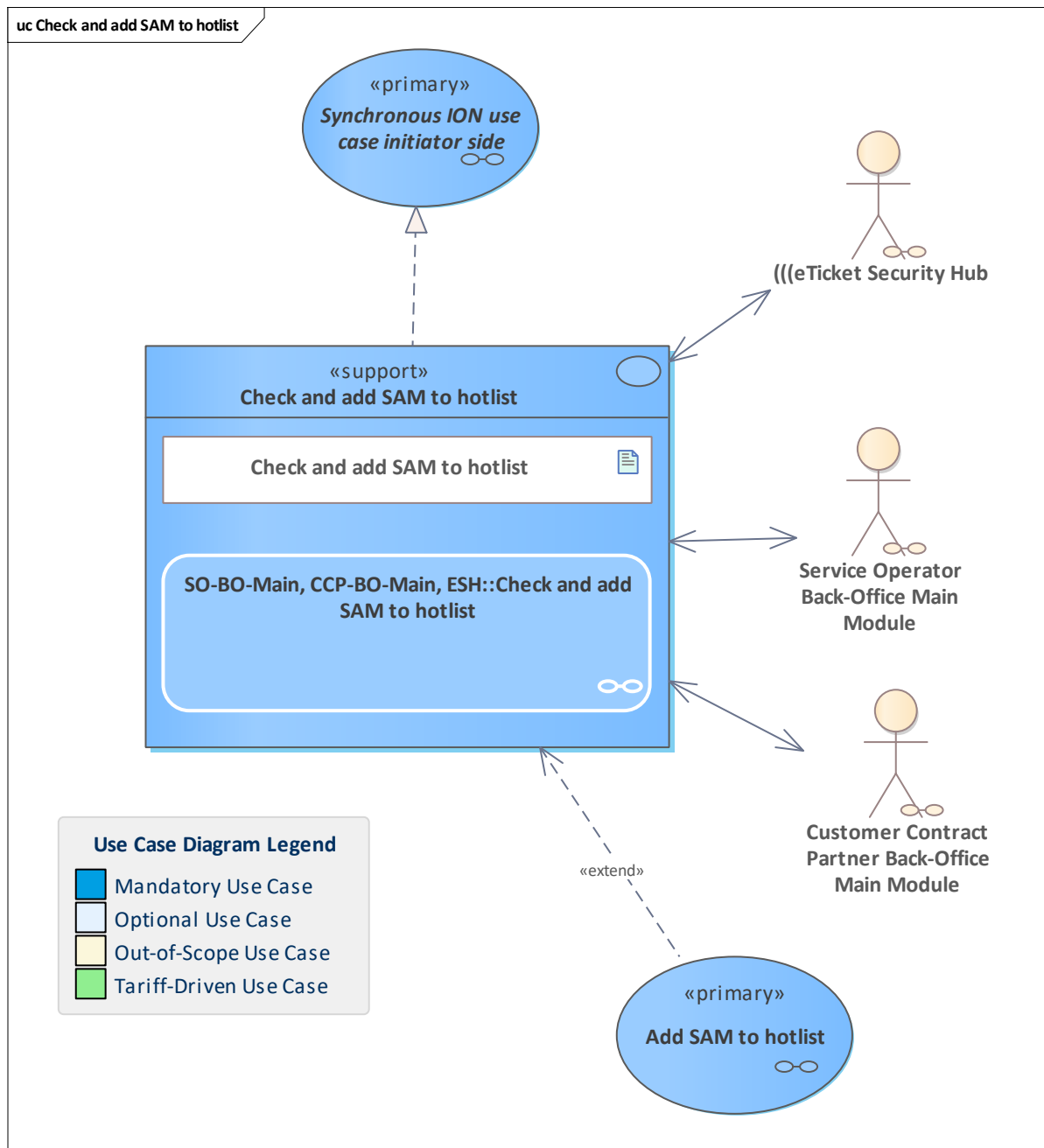
<b>Use Case</b>	<a href="#">Handle SAM hotlisting demand</a>
<b>Description</b>	<p>The owner (SO or CCP) of a SAM to be hotlisted checks the hotlisting demand. If the result of the check is such that a hotlisting is required, it will inform the hotlist service system to have the SAM added to the SAM hotlist. Otherwise, the demand will be rejected and there is no need to communicate with the hotlist service system.</p> <p>Please note that if there is more than one demand for hotlisting the same SAM, this has to be considered in the check for a required hotlisting but will not result in an exception. Especially for the monitoring of a third-party system, it must be possible to demand hotlisting even if the same object was demanded for hotlisting in the past.</p> <p>Please note that the scheme manager is allowed to hotlist SAMs, for example in case of a hotlisted organisation or as an escalation instance.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>



	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Check and add SAM to hotlist</a>
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Demand SAM hotlisting : demandSamHotlisting</a>
<b>Outputs</b>	<a href="#">Demand SAM hotlisting response : demandSamHotlistingResponse</a>
<b>Error Cases</b>	<a href="#">E_CO_APP_INSTANCE_ID_UNKNOWN</a> <a href="#">E_CO_WRONG_SECURITY_LEVEL</a> <a href="#">Demand SAM hotlisting exception : demandSamHotlistingException</a>
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle SAM hotlisting demand</a>



## 6.2.2 Check and add SAM to hotlist

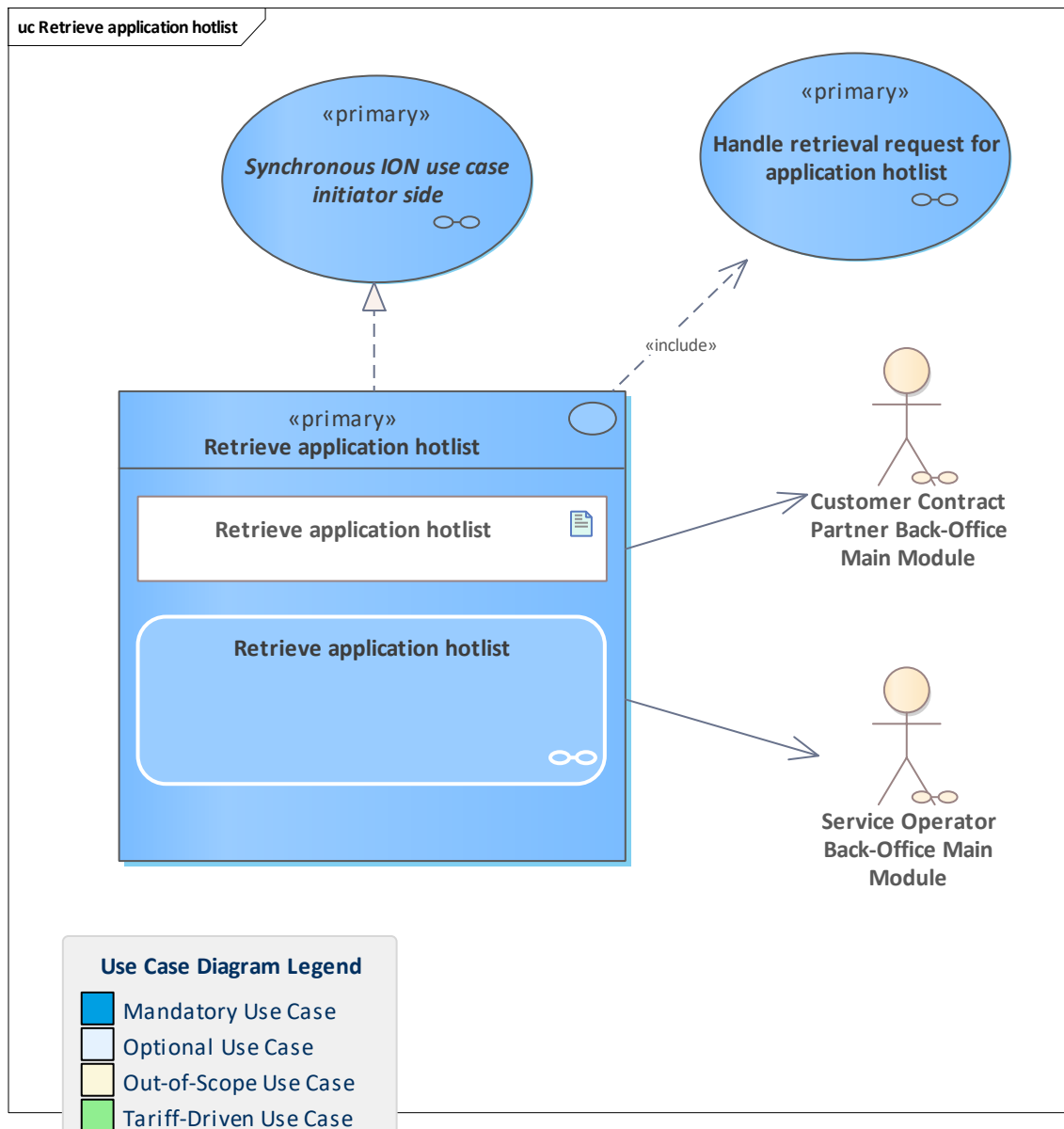


Use Case	Check and add SAM to hotlist
Description	<p>Supporting use case for the Scheme Manager, SO or CCP.</p> <p>The parameters required to add a SAM to the hotlist are checked, especially if another system has already requested the SAM to be added to the hotlist.</p> <p>In addition, the SAM owner may add specific counter information that is not available to a third party system.</p> <p>The hotlist service system is requested to add the SAM to the SAM</p>



	hotlist.
<b>Initiating Actor</b>	<a href="#">(((eTicket Security Hub</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">(((eTicket Security Hub</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Add SAM to hotlist</a>
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Blocking Reason : BlockingReason</a> <a href="#">SAM action counter : ActionCounter</a> <a href="#">SAM ID : AppInstanceId</a> <a href="#">SAM entitlement issuance counter : EntitlementIssuanceCounter</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main, ESH::Check and add SAM to hotlist</a>

## 6.2.3 Retrieve application hotlist

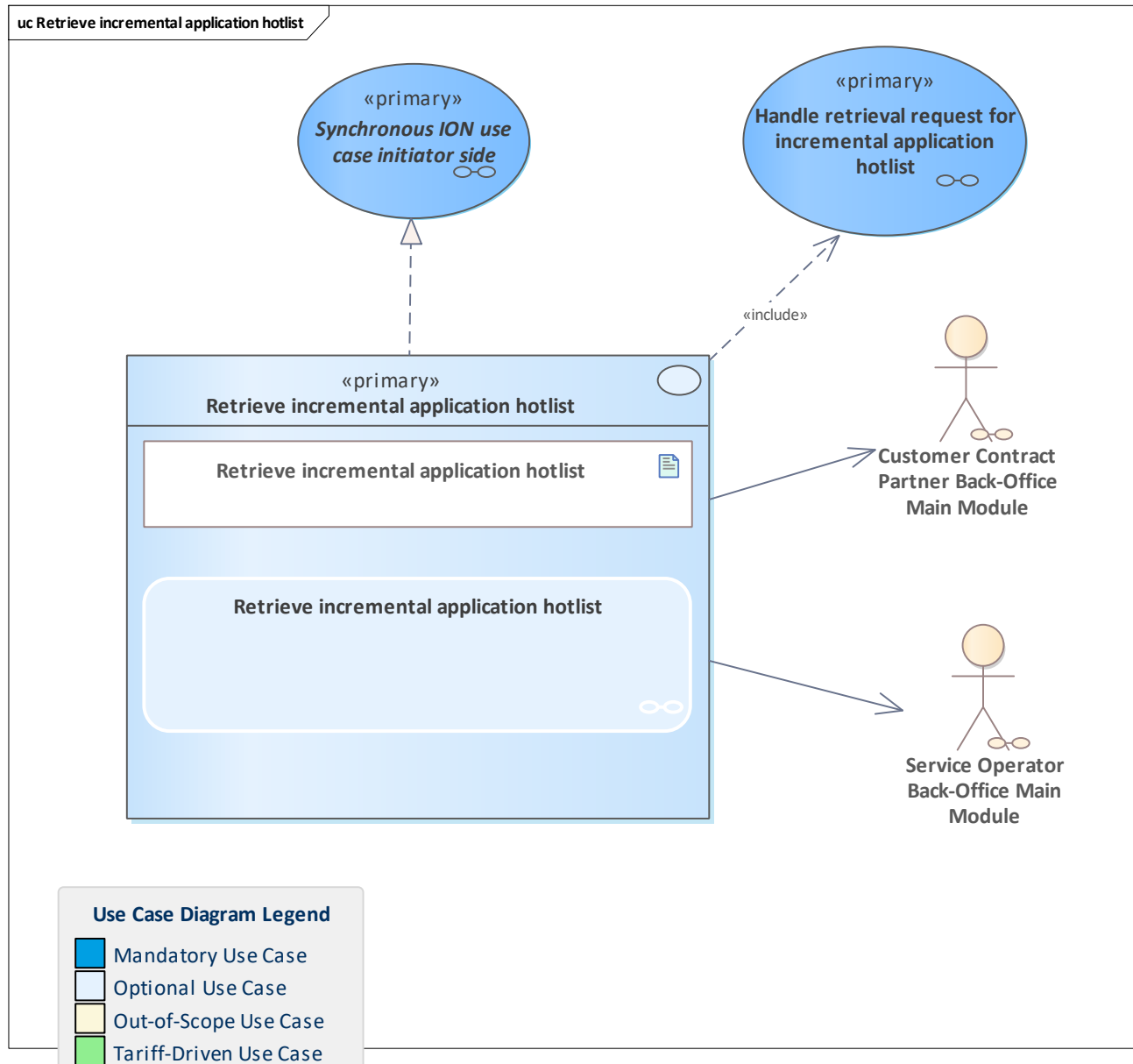


<b>Use Case</b>	<a href="#">Retrieve application hotlist</a>
<b>Description</b>	The SO and CCP want to update their application hotlist inventory by retrieving the application hotlist from the hotlist service system. Please note that the application hotlist can be retrieved as an incremental or a total hotlist.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for application hotlist</a>



<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a>
<b>Outputs</b>	<a href="#">Application hotlist : ApplicationHotlist</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Retrieve application hotlist</a>

## 6.2.4 Optional: Retrieve incremental application hotlist

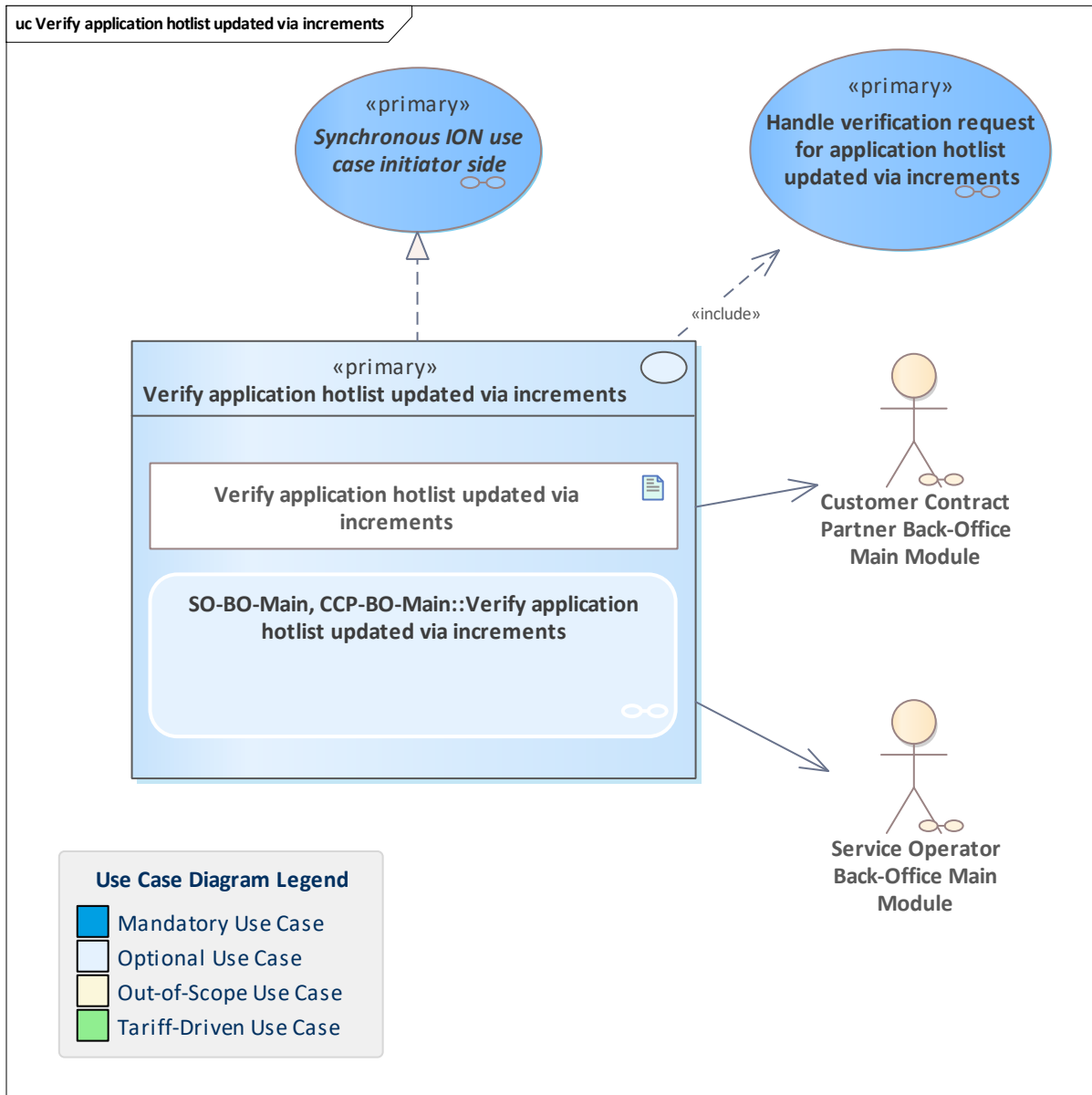


<b>Use Case</b>	<a href="#">Retrieve incremental application hotlist</a>
<b>Description</b>	The SO and CCP want to update their application hotlist inventory by retrieving the current incremental application hotlist from the hotlist service system. Please note that the application hotlist can be retrieved as an incremental or a total hotlist.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	



<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for incremental application hotlist</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a>
<b>Outputs</b>	<a href="#">Updated application hotlist : ApplicationHotlist</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Retrieve incremental application hotlist</a>

## 6.2.5 Optional: Verify application hotlist updated via increments



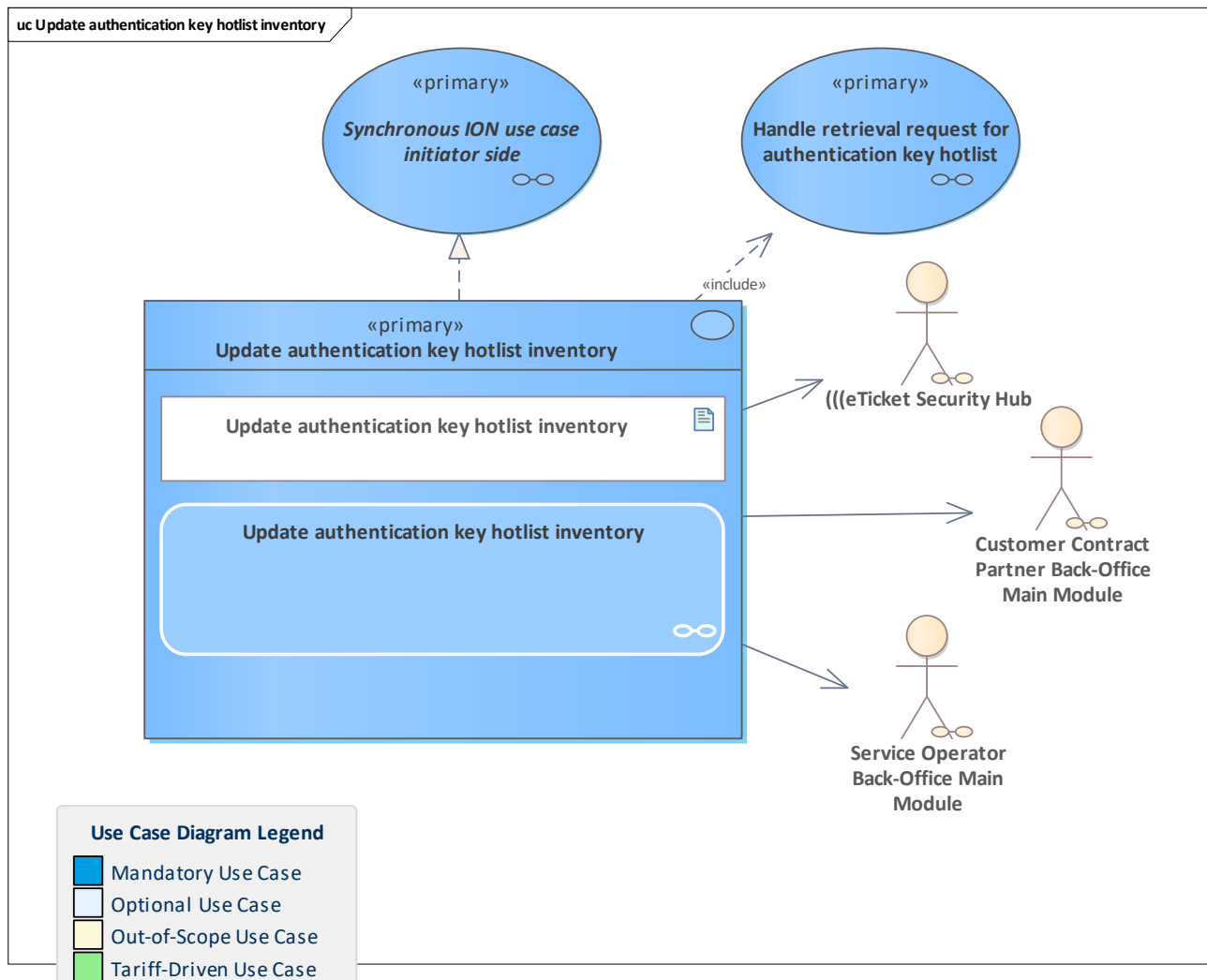
<b>Use Case</b>	<u>Verify application hotlist updated via increments</u>
<b>Description</b>	<p>After updating the application hotlist inventory via the last incremental application hotlist, the participant must ensure that the updated hotlist inventory is the same as the hotlist inventory of the hotlist service system.</p> <p>For that reason, participants compute the checksum of all the application instance IDs in their inventory and send it to the hotlist service system to verify the value of the checksum. See also <a href="#">Checksum calculation for hotlist and action list verification</a> and <a href="#">Example calculation for an application hotlist inventory</a>.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u>



	<a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle verification request for application hotlist updated via increments</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a> <a href="#">Updated application hotlist : ApplicationHotlist</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Verify application hotlist updated via increments</a>



## 6.2.6 Update authentication key hotlist inventory

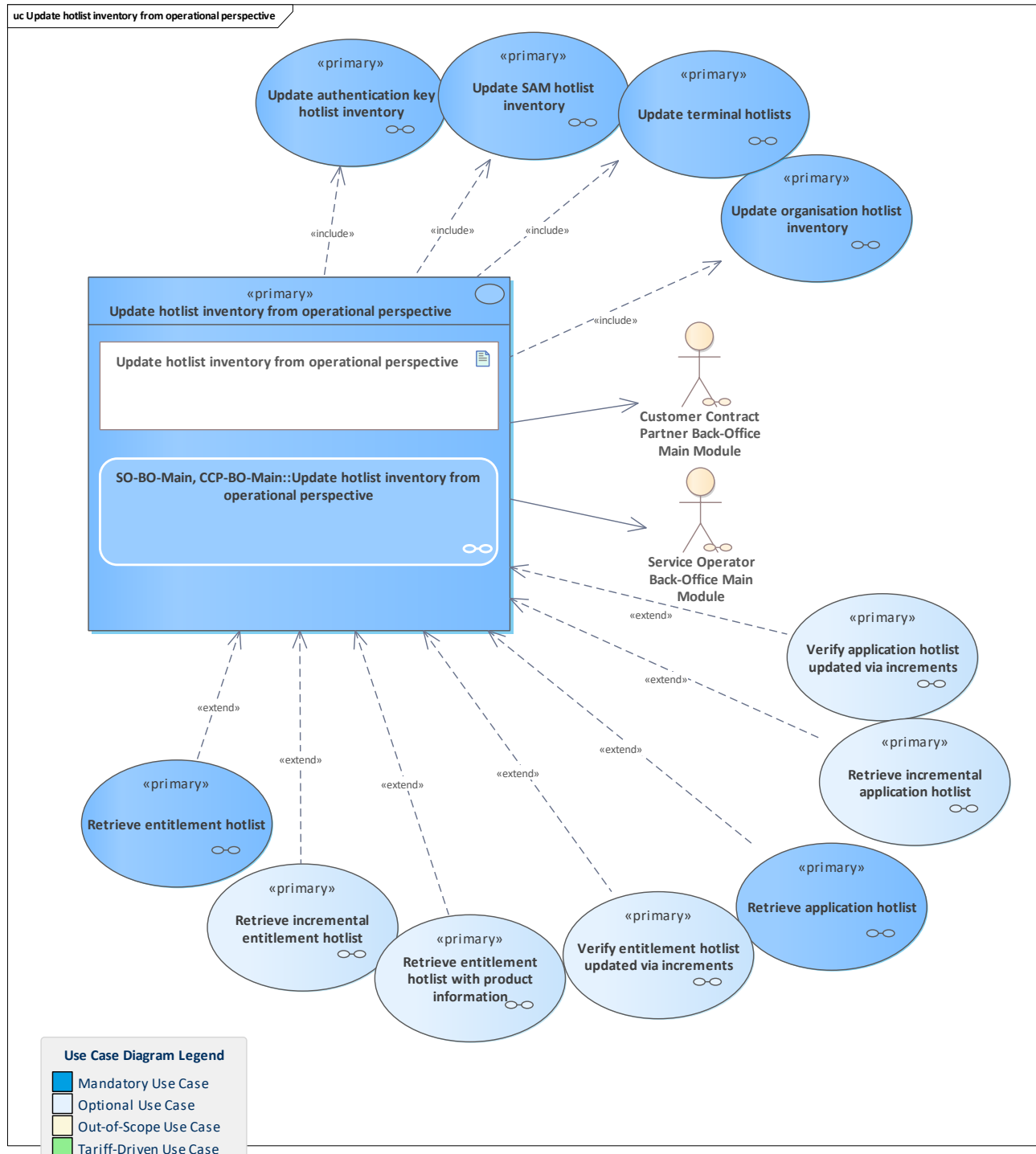


<b>Use Case</b>	<a href="#">Update authentication key hotlist inventory</a>
<b>Description</b>	The SO, CCP and the scheme manager's ESH want to update the authentication key hotlist inventory by retrieving the current authentication key hotlist from the hotlist service system and processing it into their authentication key hotlist inventory.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">((eTicket Security Hub</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for authentication key hotlist</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	



<b>Inputs</b>	
<b>Outputs</b>	<a href="#">Updated authentication key hotlist inventory</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">Update authentication key hotlist inventory</a>

## 6.2.7 Update hotlist inventory from operational perspective

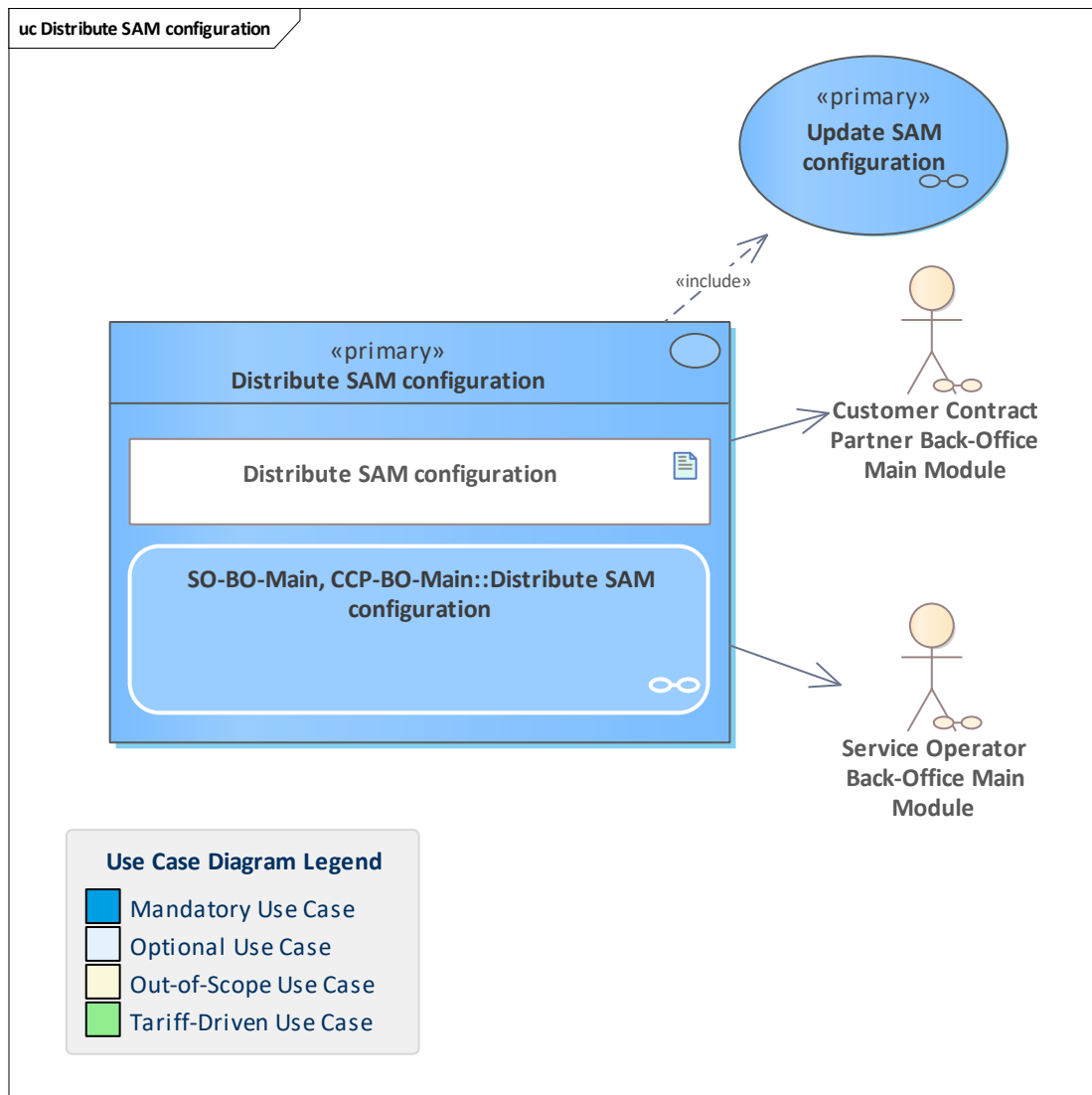


<b>Use Case</b>	<u>Update hotlist inventory from operational perspective</u>
<b>Description</b>	A CCP or SO wants to retrieve the currently available hotlists and distribute them to its terminals to inspect accessed entitlements and applications, as well as to deactivate a terminal if its SAM has been hotlisted.



	<p>The available hotlists to be updated and distributed are:</p> <ul style="list-style-type: none"><li>• Application hotlist:  either total application hotlist or incremental application hotlist</li><li>• Entitlement hotlist:  either total entitlement hotlist or incremental entitlement hotlist or total entitlement hotlist with product information</li><li>• SAM hotlist</li><li>• Organisation hotlist</li><li>• Authentication key hotlist</li></ul>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Retrieve incremental application hotlist</a> / <a href="#">Verify application hotlist updated via increments</a> / <a href="#">Retrieve application hotlist</a> / <a href="#">Verify entitlement hotlist updated via increments</a> / <a href="#">Retrieve entitlement hotlist</a> / <a href="#">Retrieve incremental entitlement hotlist</a> / <a href="#">Retrieve entitlement hotlist with product information</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Update SAM hotlist inventory</a> / <a href="#">Update authentication key hotlist inventory</a> / <a href="#">Update organisation hotlist inventory</a> / <a href="#">Update terminal hotlists</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Update hotlist inventory from operational perspective</a>

## 6.2.8 Distribute SAM configuration

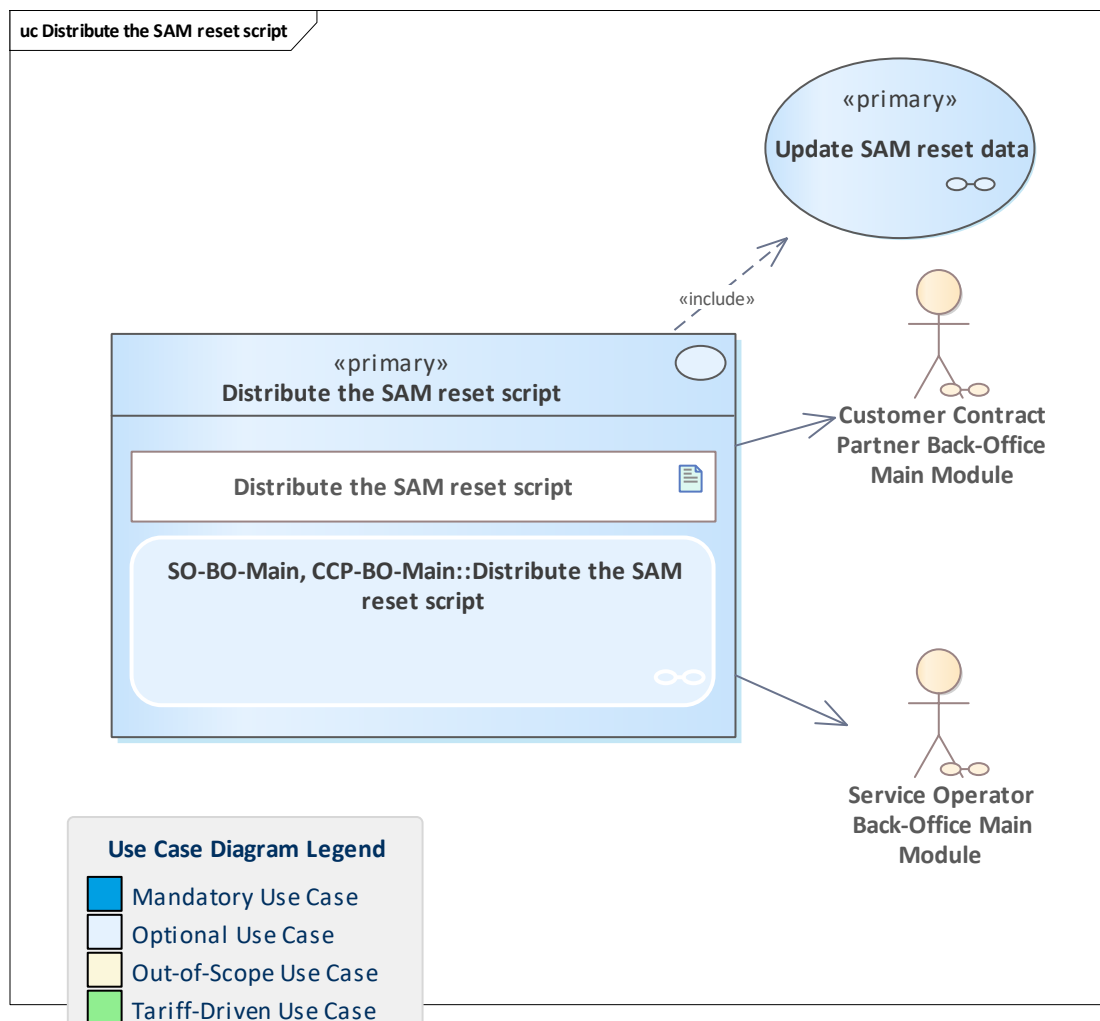


<b>Use Case</b>	<a href="#">Distribute SAM configuration</a>
<b>Description</b>	<p>Required SAM configuration data was received from the ESH. The back-office system distributes the script for each new SAM configuration.</p> <p>In this use case, it is assumed that only one SAM configuration per terminal is required for the sake of simplicity.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Update SAM configuration</a>
<b>Linked Use Cases</b>	



<b>(Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">List of SAM configuration</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Distribute SAM configuration</a>

## 6.2.9 Optional: Distribute the SAM reset script



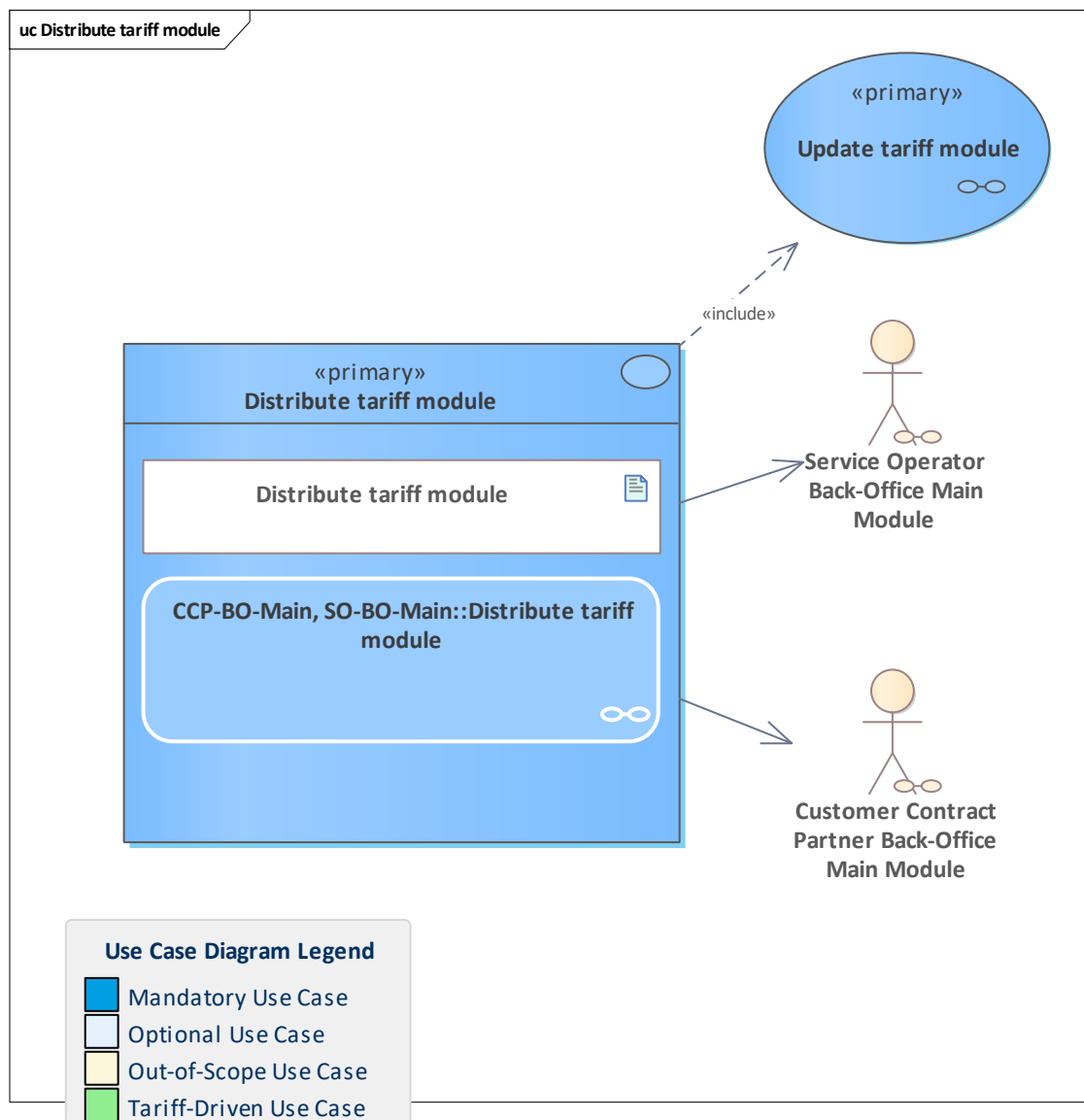
<b>Use Case</b>	<a href="#">Distribute the SAM reset script</a>
<b>Description</b>	<p>Required SAM reset data was received from the ESH. The back-office system distributes the script for each SAM that needs to be reset.</p> <p>In this use case it is assumed that only one SAM reset per terminal is required for the sake of simplicity.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Update SAM reset data</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	



<b>Inputs</b>	<a href="#">List of SAM reset data</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Distribute the SAM reset script</a>



## 6.2.10 Distribute tariff module

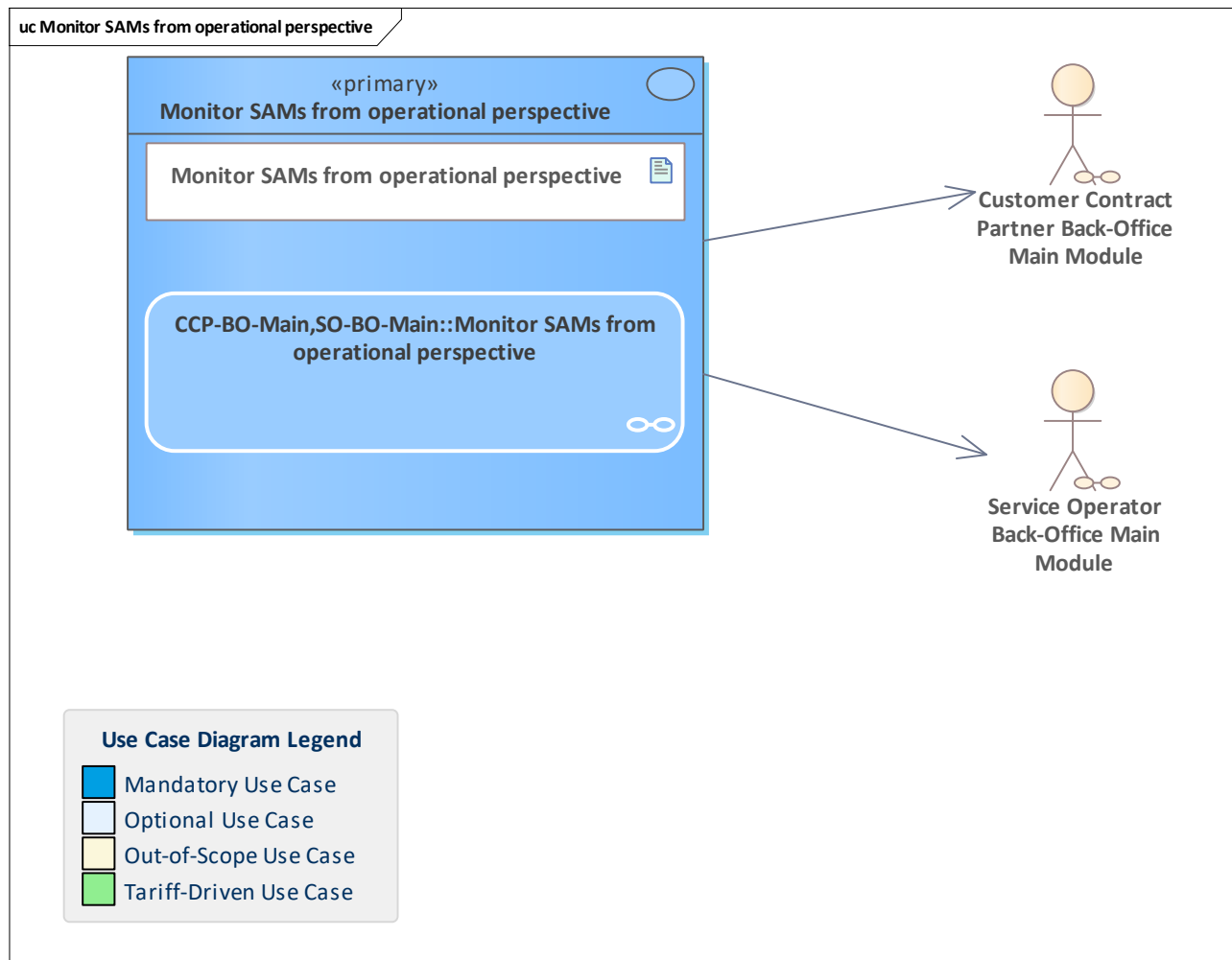


<b>Use Case</b>	<u>Distribute tariff module</u>
<b>Description</b>	<p>The CPP and SO back-office system retrieves tariff modules from POs (out of scope) and stores them in its data store.</p> <p>The SO and CCP create one tariff module for terminals based on the tariff modules received from the POs. The SO and CCP distributes this tariff module for its terminals. This process needs to run periodically to keep the tariff module up to date.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Service Operator Back-Office Main Module</u> <u>Customer Contract Partner Back-Office Main Module</u>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	



<b>Linked Use Cases (Includes)</b>	<a href="#">Handle LDAP search request</a> / <a href="#">Update tariff module</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main, SO-BO-Main::Distribute tariff module</a>

## 6.2.11 Monitor SAMs from operational perspective



<b>Use Case</b>	<a href="#">Monitor SAMs from operational perspective</a>
<b>Description</b>	The own SAMs need to be monitored from the operational perspective. All entitlement issuances and action authorisations have to be documented using notifications.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	



<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#"><u>CCP-BO-Main,SO-BO-Main::Monitor SAMs from operational perspective</u></a>

## 7 Basic Bundle Terminal Operator System - Extended Logging

This optional functionality bundle allows the processing of extended logging notifications for an application or an entitlement. The extended logging can also be used for static entitlements.

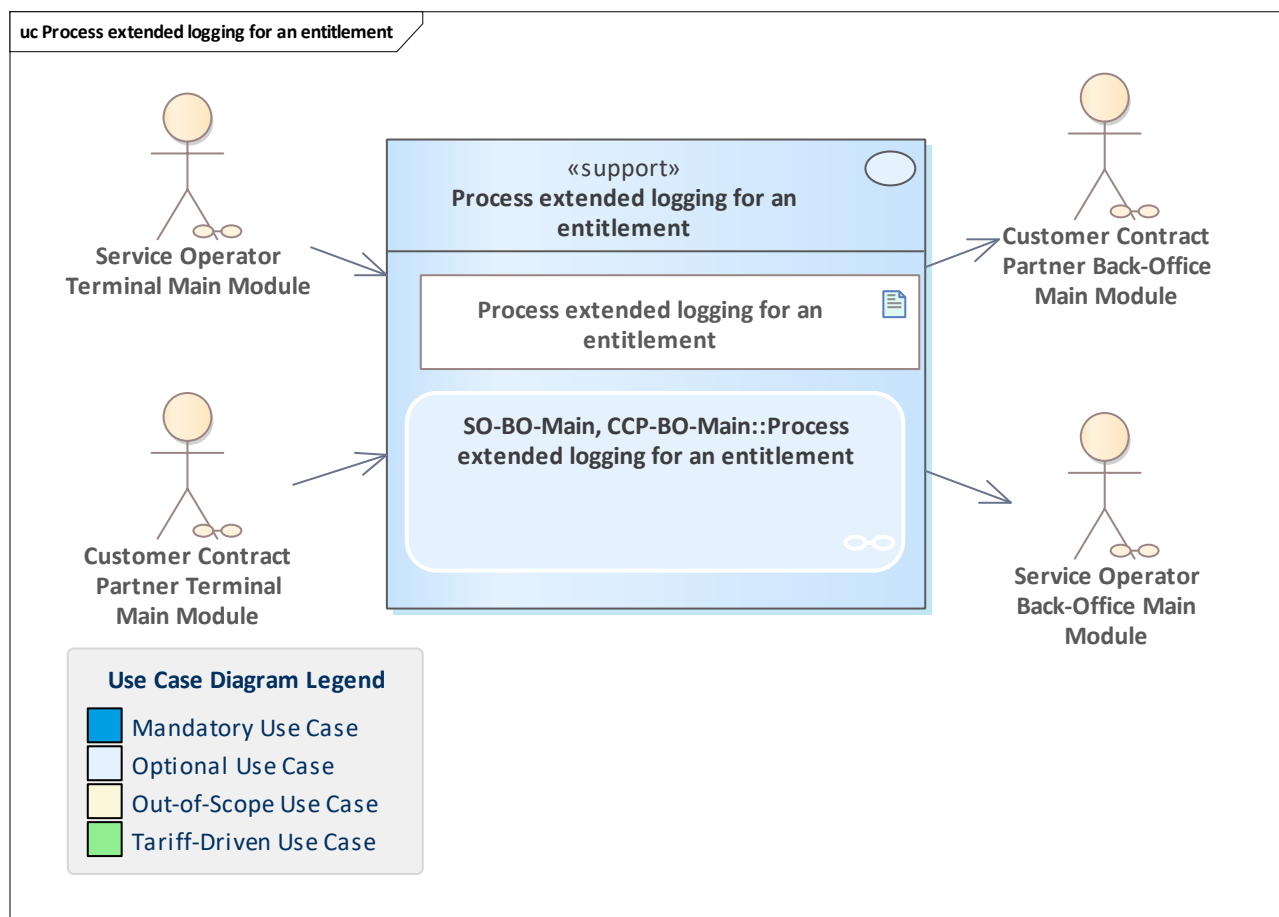
### 7.1 Overview

Optional: Process extended logging for an entitlement

Optional: Process extended logging for an application

### 7.2 Use Cases

#### 7.2.1 Optional: Process extended logging for an entitlement

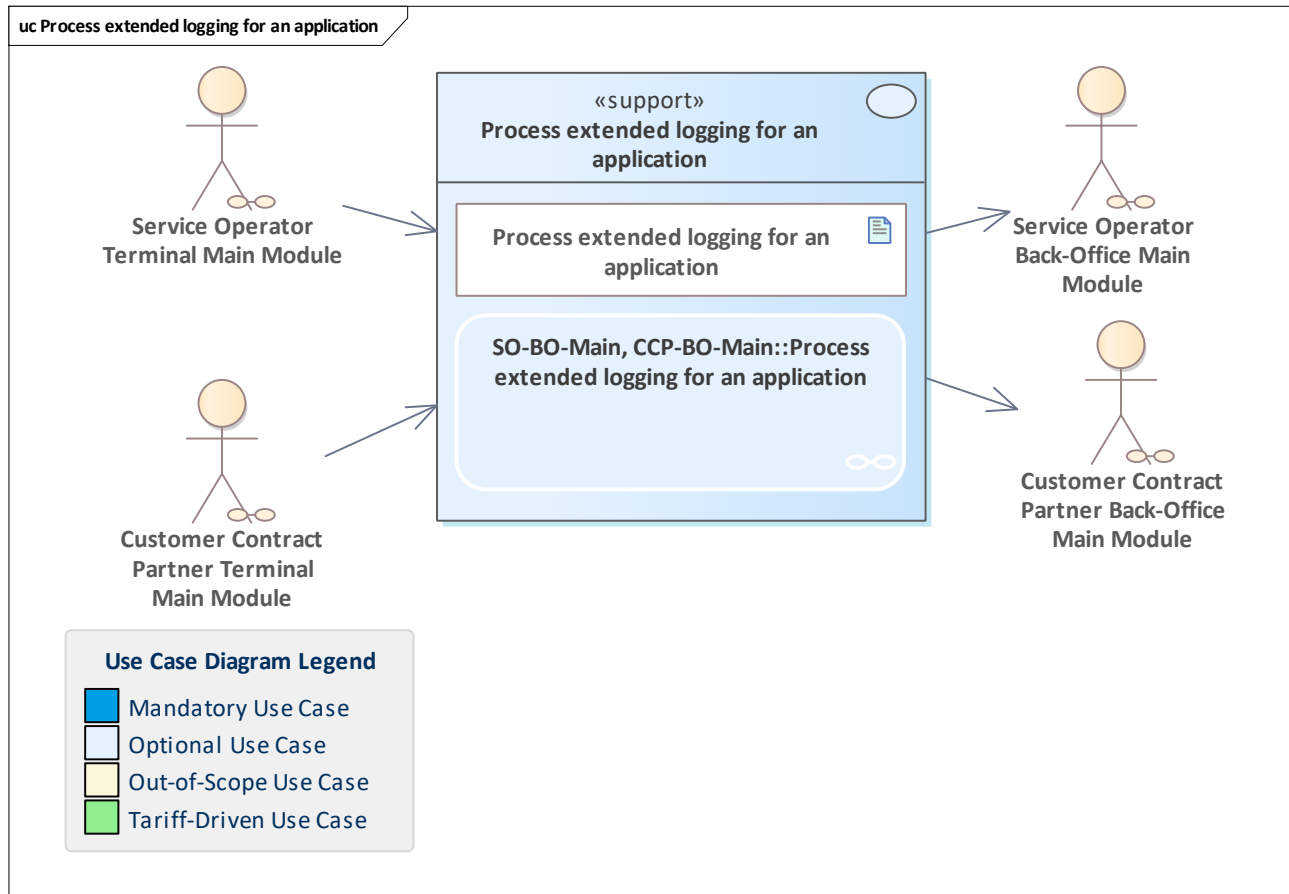


<b>Use Case</b>	<a href="#">Process extended logging for an entitlement</a>
<b>Description</b>	Extended logging for an entitlement is received from a terminal by the back-office system and registered.



<b>Initiating Actor</b>	<a href="#">Service Operator Terminal Main Module</a> <a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Log for invalid entitlement from terminal : tLogInvalidEntitlement</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Process extended logging for an entitlement</a>

## 7.2.2 Optional: Process extended logging for an application



<b>Use Case</b>	<a href="#">Process extended logging for an application</a>
<b>Description</b>	Extended logging for an application is received from a terminal by the back-office system and registered.
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a> <a href="#">Service Operator Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Log for invalid application from terminal : tLogInvalidApplication</a>
<b>Outputs</b>	



<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Process extended logging for an application</a>



## 8 Basic Bundle Terminal Operator System - UM with Application

Functionality bundle that covers all the basic use cases required for a terminal operator system that works with user media with an application.

### 8.1 Overview

Handle defective user medium with application

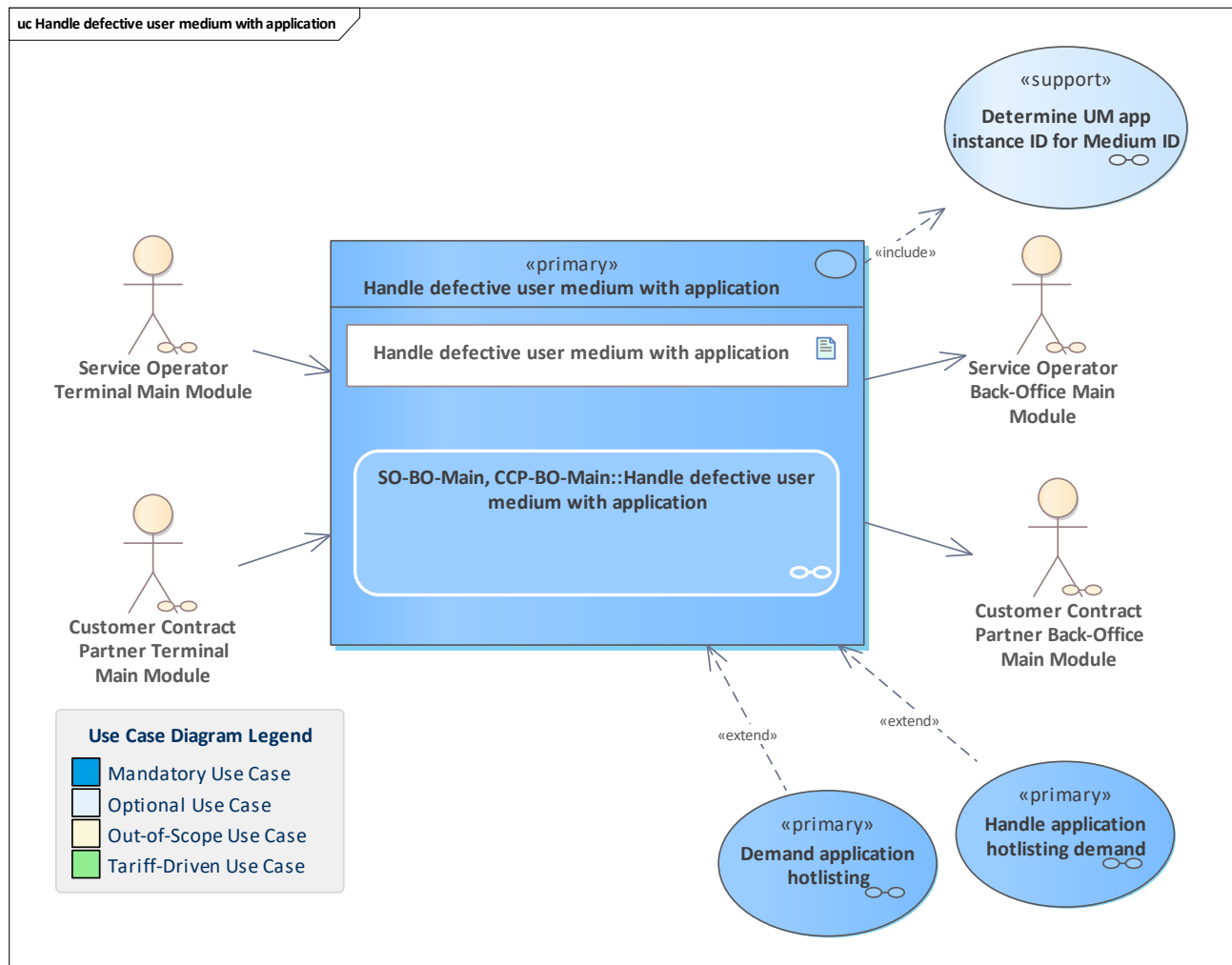
Optional: Determine UM app instance ID for Medium ID

Handle application blocked notification from operational perspective

Handle entitlement blocked notification from operational perspective

### 8.2 Use Cases

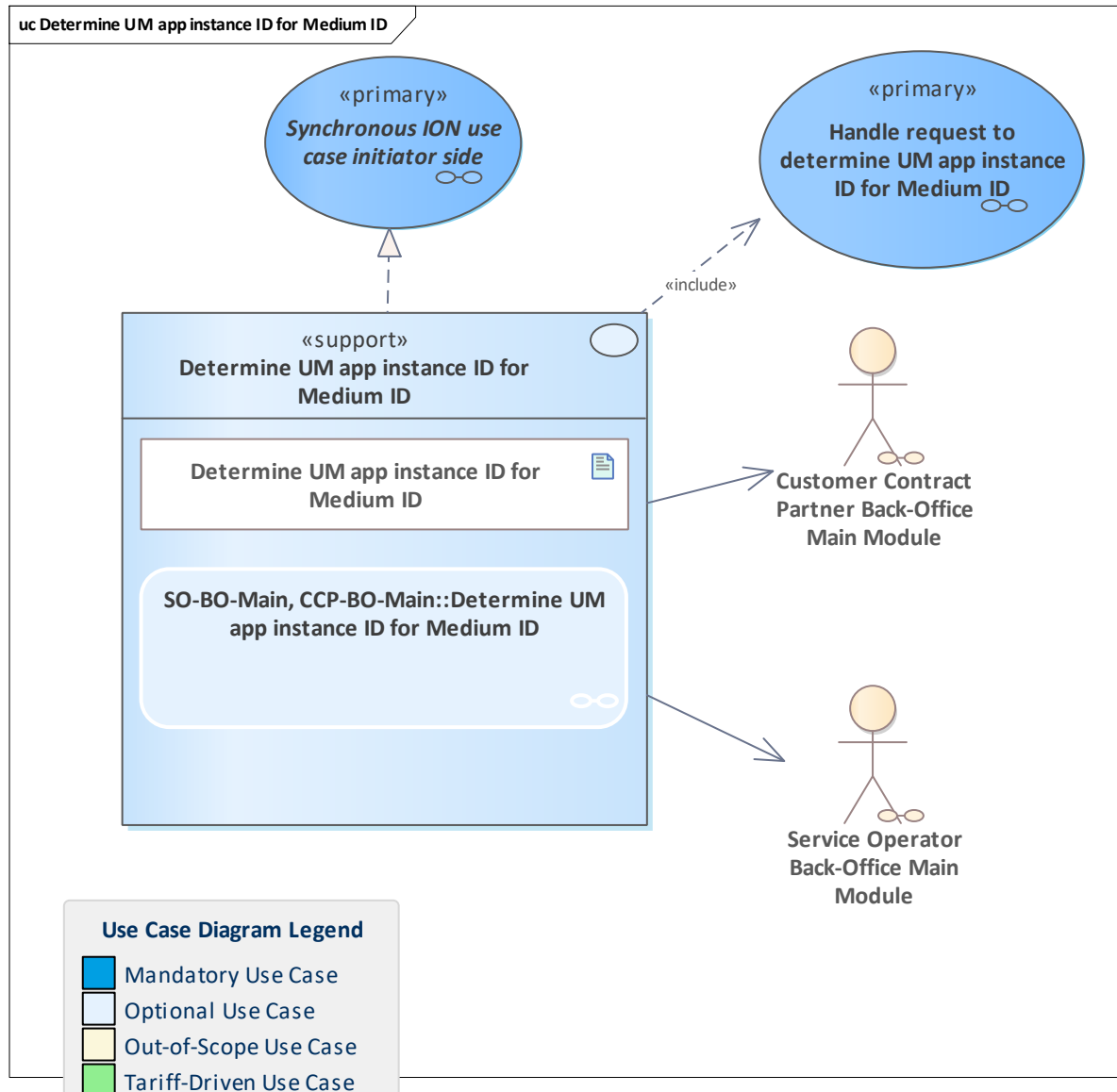
#### 8.2.1 Handle defective user medium with application





<b>Use Case</b>	<a href="#">Handle defective user medium with application</a>
<b>Description</b>	<p>The terminal operator (CCP or SO) back-office system receives the log data of the terminal which detected the defective user medium. The message also contains the medium ID which can be the application instance ID. If the medium ID is proprietary, the application instance ID must be determined for the received medium ID.</p> <p>The application of the application instance ID must be hotlisted. Depending on the role in the current process (pCCP versus sCCP/SO) either a hotlist demand is sent to the application instance owner, or the pCCP communicates directly with the hotlist service.</p>
<b>Initiating Actor</b>	<a href="#">Service Operator Terminal Main Module</a> <a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Handle application hotlisting demand</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle application hotlisting demand</a> / <a href="#">Determine UM app instance ID for Medium ID</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Log defective user medium with application from terminal : tLogDefectiveUserMediumWithApplication</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle defective user medium with application</a>

## 8.2.2 Optional: Determine UM app instance ID for Medium ID

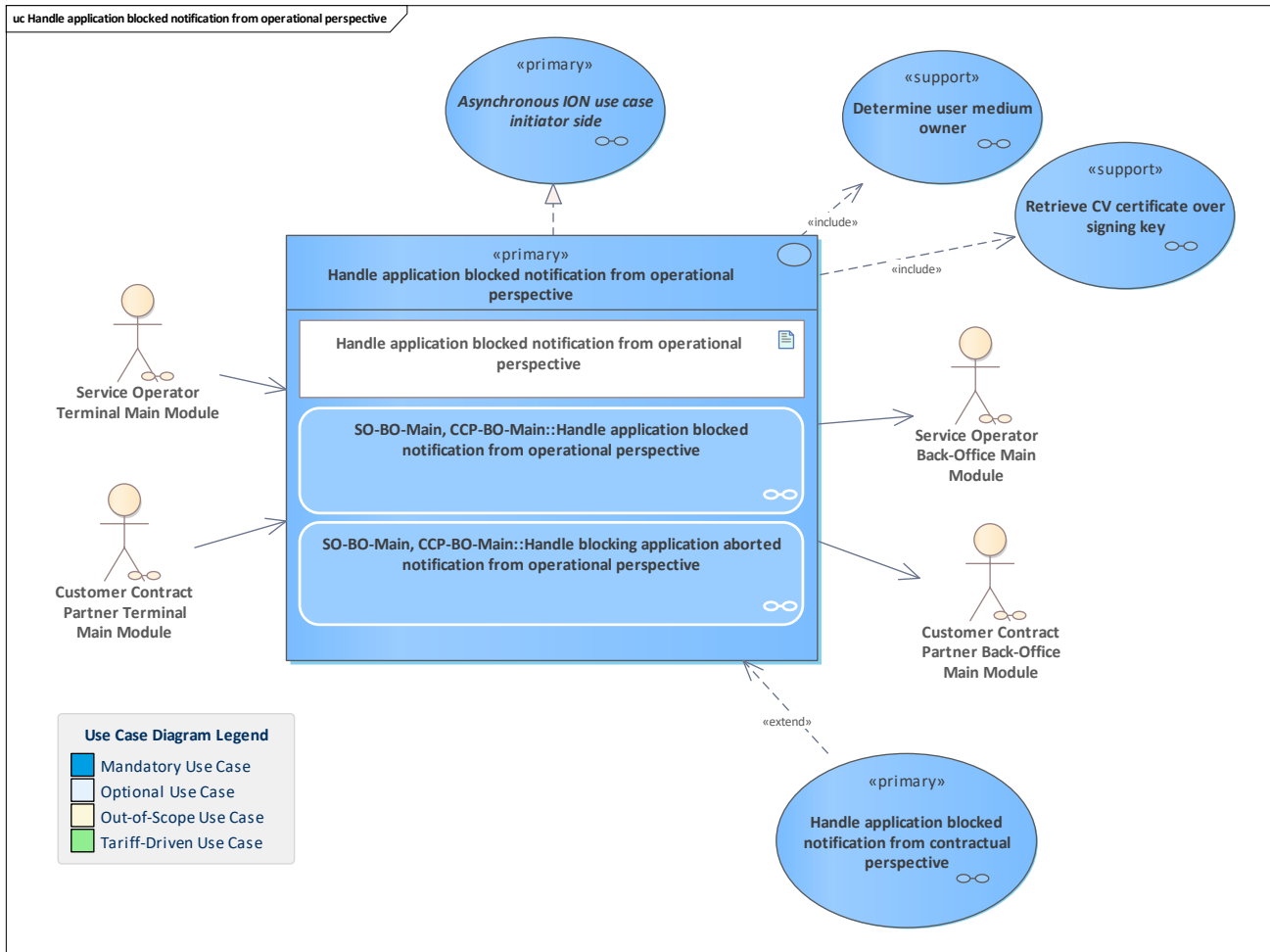


<b>Use Case</b>	<a href="#">Determine UM app instance ID for Medium ID</a>
<b>Description</b>	Determine the user medium application instance ID for a given medium ID using the service provided by the ESH. If the application instance ID is already printed on all user media of the transport company, then this use case is optional.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle request to determine UM app instance ID for Medium ID</a>



<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Medium ID : MediumId</a>
<b>Outputs</b>	<a href="#">UM app Instance ID : AppInstanceId</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Determine UM app instance ID for Medium ID</a>

## 8.2.3 Handle application blocked notification from operational perspective

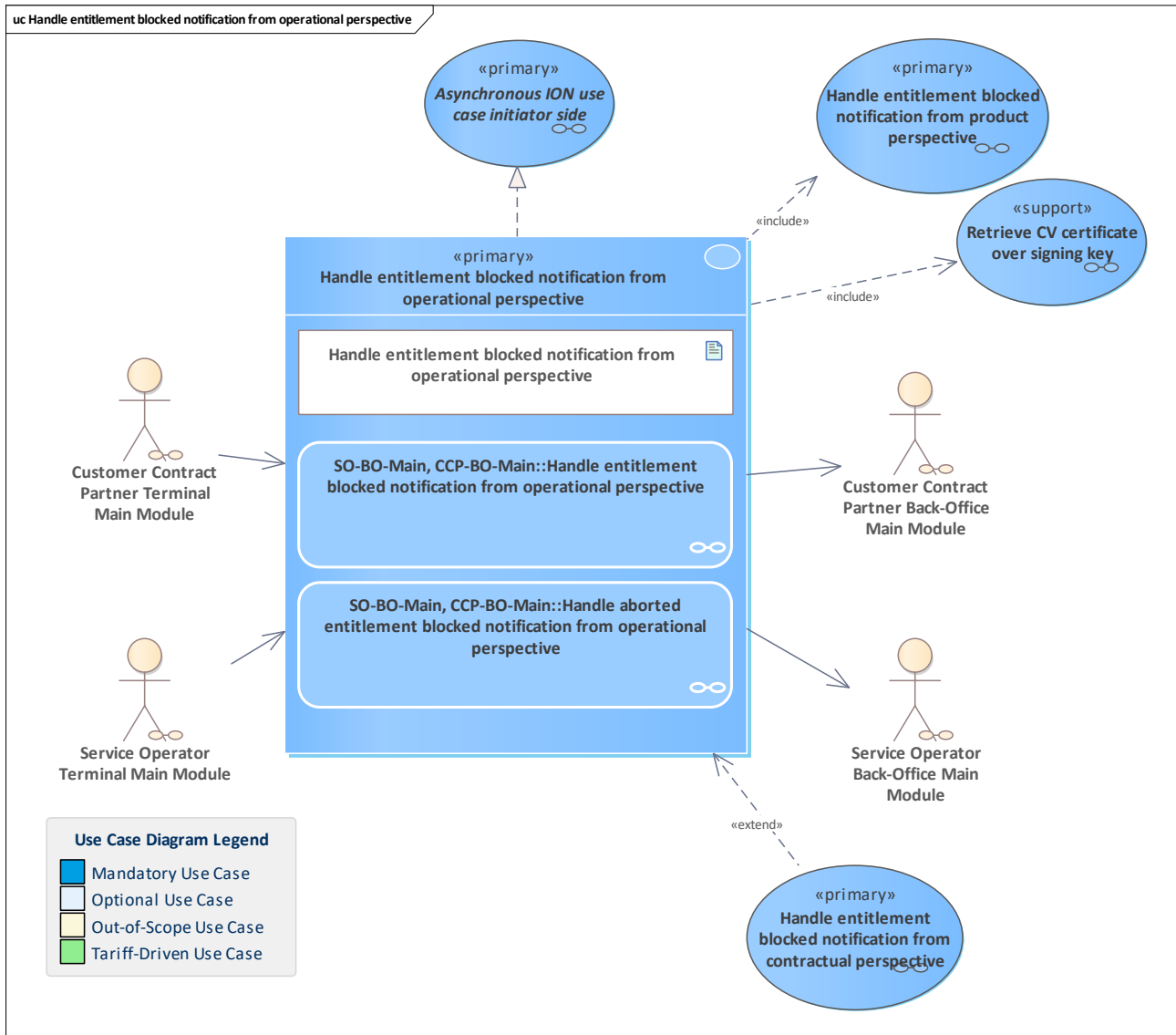


Use Case	<a href="#">Handle application blocked notification from operational perspective</a>
Description	<p>A terminal has blocked the application in a user medium with an application.</p> <p>In this use case, the SO and/or CCP back-office system receives the notification from the terminal and runs certain checks from the issuer perspective, such as the signature verification of the block application attestation.</p> <p>The pCCP of the application is informed if the current operator is a SO or sCCP.</p> <p>In the case of action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The SO or the CCP back-office system registers the abortion for internal monitoring.</p>
Initiating Actor	<a href="#">Customer Contract Partner Terminal Main Module</a> <a href="#">Service Operator Terminal Main Module</a>
Reacting Actor	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
Preconditions	
Postconditions	



<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle application blocked notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Determine user medium owner</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Application blocked notification from terminal : tNotifyApplicationBlocked</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle application blocked notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify application blocking aborted from terminal : tNotifyApplicationBlockingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle blocking application aborted notification from operational perspective</a>

## 8.2.4 Handle entitlement blocked notification from operational perspective



<b>Use Case</b>	<a href="#">Handle entitlement blocked notification from operational perspective</a>
<b>Description</b>	<p>The entitlement blocked notification is sent by the terminal to the SO or CCP back-office system. The notification will be checked.</p> <p><u>If the pCCP blocked its own entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO</li> <li>request the hotlist service system to remove the entitlement from the entitlement hotlist.</li> </ul> <p><u>If the sCCP or SO blocked an entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO (and the PO will forward it later to the pCCP)</li> </ul>



	In case of action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The SO or CCP back-office system registers the abortion for internal monitoring and data consistency.
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a> <a href="#">Service Operator Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Service Operator Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle entitlement blocked notification from contractual perspective</a> / <a href="#">Handle entitlement blocked notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement blocked notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notify entitlement blocked from terminal : tNotifyEntitlementBlocked</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle entitlement blocked notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify entitlement blocked aborted from terminal : tNotifyEntitlementBlockingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">SO-BO-Main, CCP-BO-Main::Handle aborted entitlement blocked notification from operational perspective</a>





## 9 Basic Bundle CCP-System - Foundation

Functionality bundle that covers all the basic use cases required for the CCP back-office system, regardless of the user media employed.

### 9.1 Overview

Handle application hotlisting demand

Handle revocation for application hotlisting demand

Handle entitlement hotlisting demand

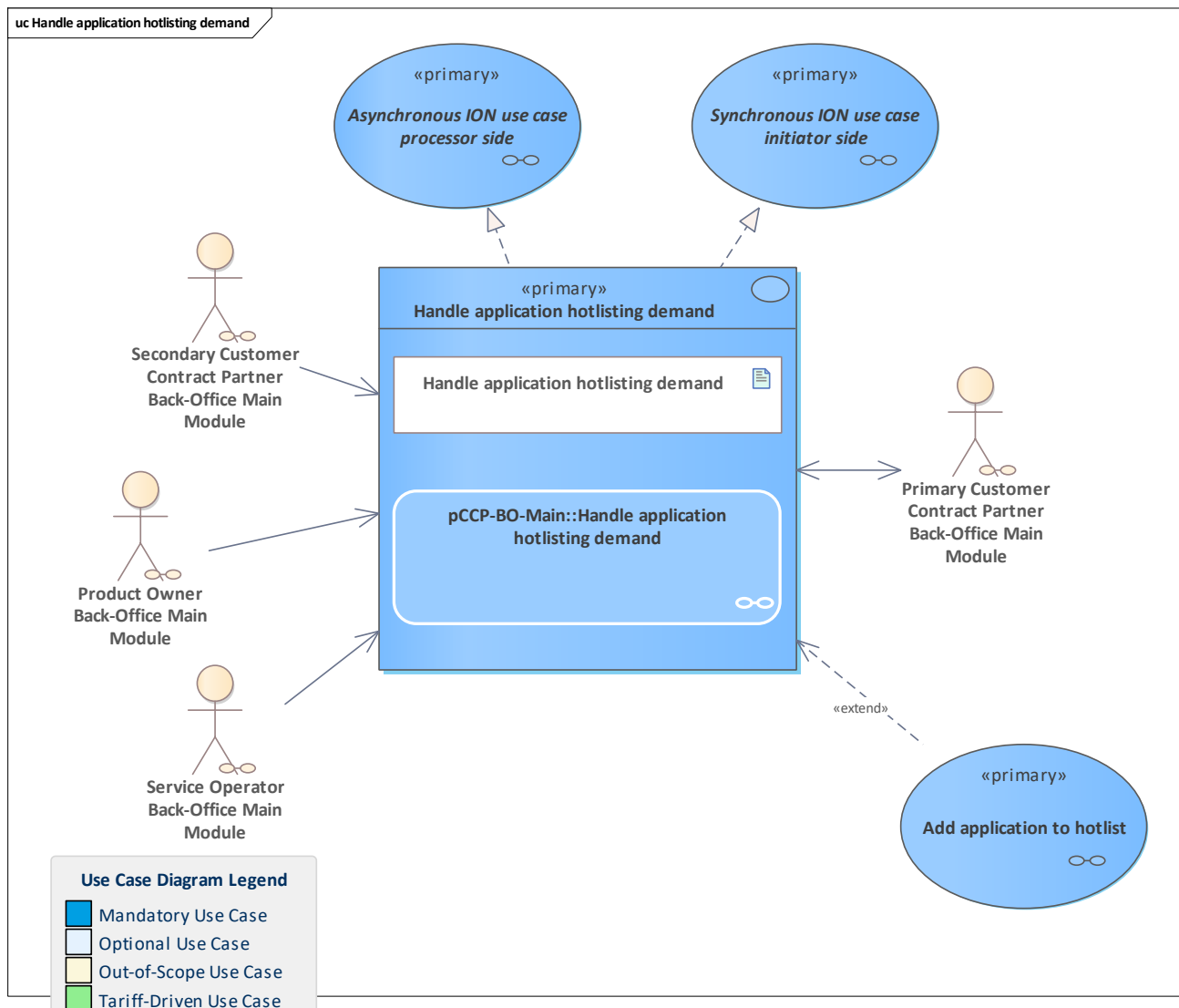
Handle revocation for entitlement hotlisting demand

Analyse entitlement history from contractual perspective

Resolve notifications with timeout warnings

### 9.2 Use Cases

#### 9.2.1 Handle application hotlisting demand

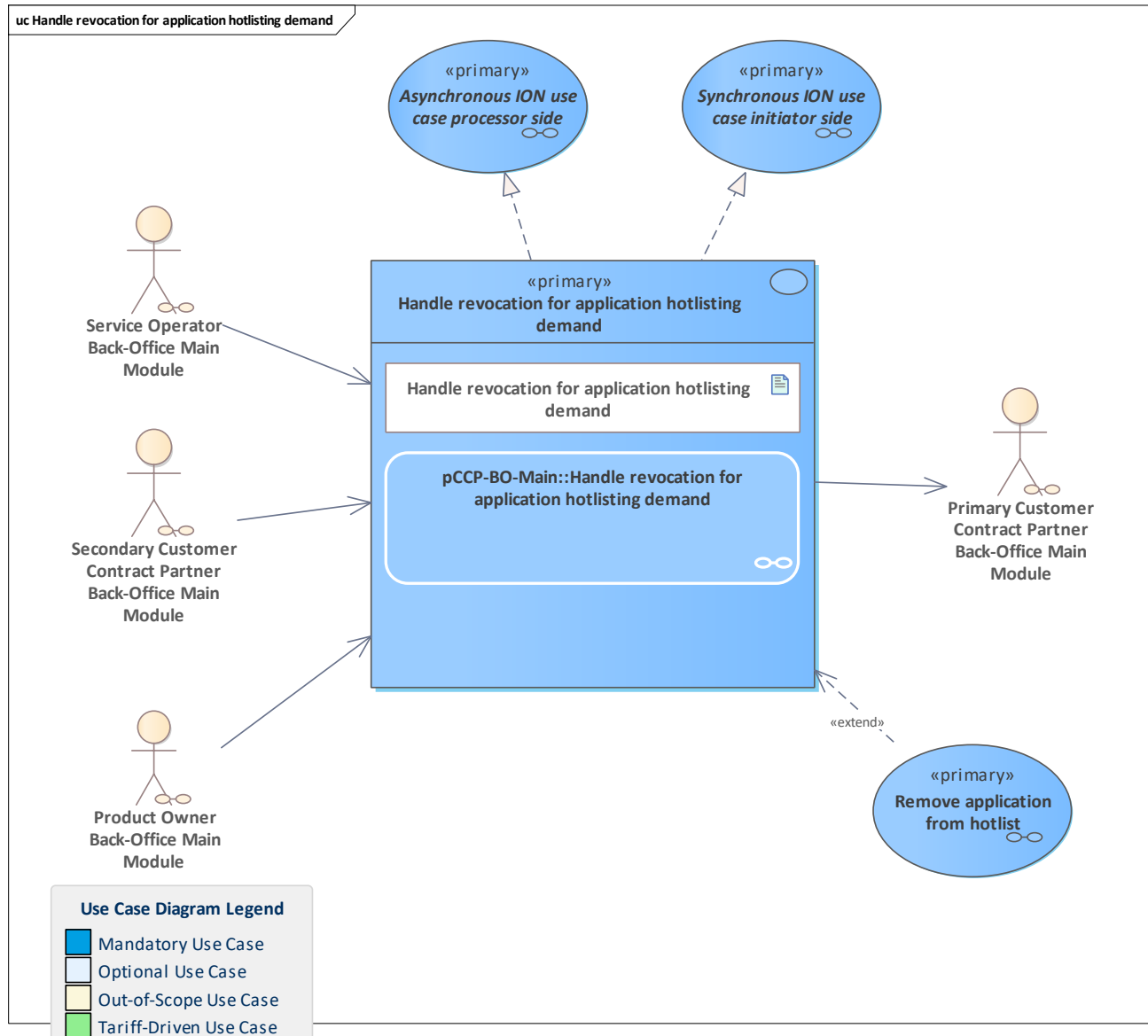


Use Case	Handle application hotlisting demand
Description	<p>The pCCP that issued the application to a customer checks the hotlisting demand.</p> <p>If the result of the check requires a hotlisting, the hotlist service system will be called to add the application to the hotlist. Otherwise, the demand will be rejected and there is no need to communicate with the hotlist service system. The original caller of the demand is informed.</p> <p>Please note that if there is more than one demand for hotlisting the same application, this has to be considered in the check for a required hotlisting but will not result in an exception. Especially for the monitoring of a third-party system, it must be possible to demand hotlisting even if the same object was demanded for hotlisting in the past.</p> <p>Several hotlisting requests for an application can be received. The final decision for a hotlisting must be decided in the pCCP, by considering all hotlisting demands, hotlisting demand revocations and hotlisting orders.</p> <p>If the application is already hotlisted, the demand will not cause a further hotlist request towards the hotlist service system.</p>



	<b>Note:</b> in the ION context, the use case is asynchronous as processor (process the demand) and a synchronous use case as initiator due to the normally performed call to the hotlist service for adding the application to the hotlist.
<b>Initiating Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a> <a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Add application to hotlist</a>
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a> / <a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Demand application hotlisting : demandApplicationHotlisting</a>
<b>Outputs</b>	<a href="#">Demand application hotlisting response : demandApplicationHotlistingResponse</a>
<b>Error Cases</b>	<a href="#">E_CO_UNKNOWN_BLOCKING_REASON</a> <a href="#">E_CO_UNKNOWN_BLOCKING_MODE</a> <a href="#">E_CCP_RECEIVER_IS_NOT_ENTITY_OWNER</a> <a href="#">E_CCP_ENTITY_ALREADY_TERMINATED</a> <a href="#">E_CO_WRONG_SECURITY_LEVEL</a> <a href="#">Demand application hotlisting exception : demandApplicationHotlistingException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle application hotlisting demand</a>

## 9.2.2 Handle revocation for application hotlisting demand

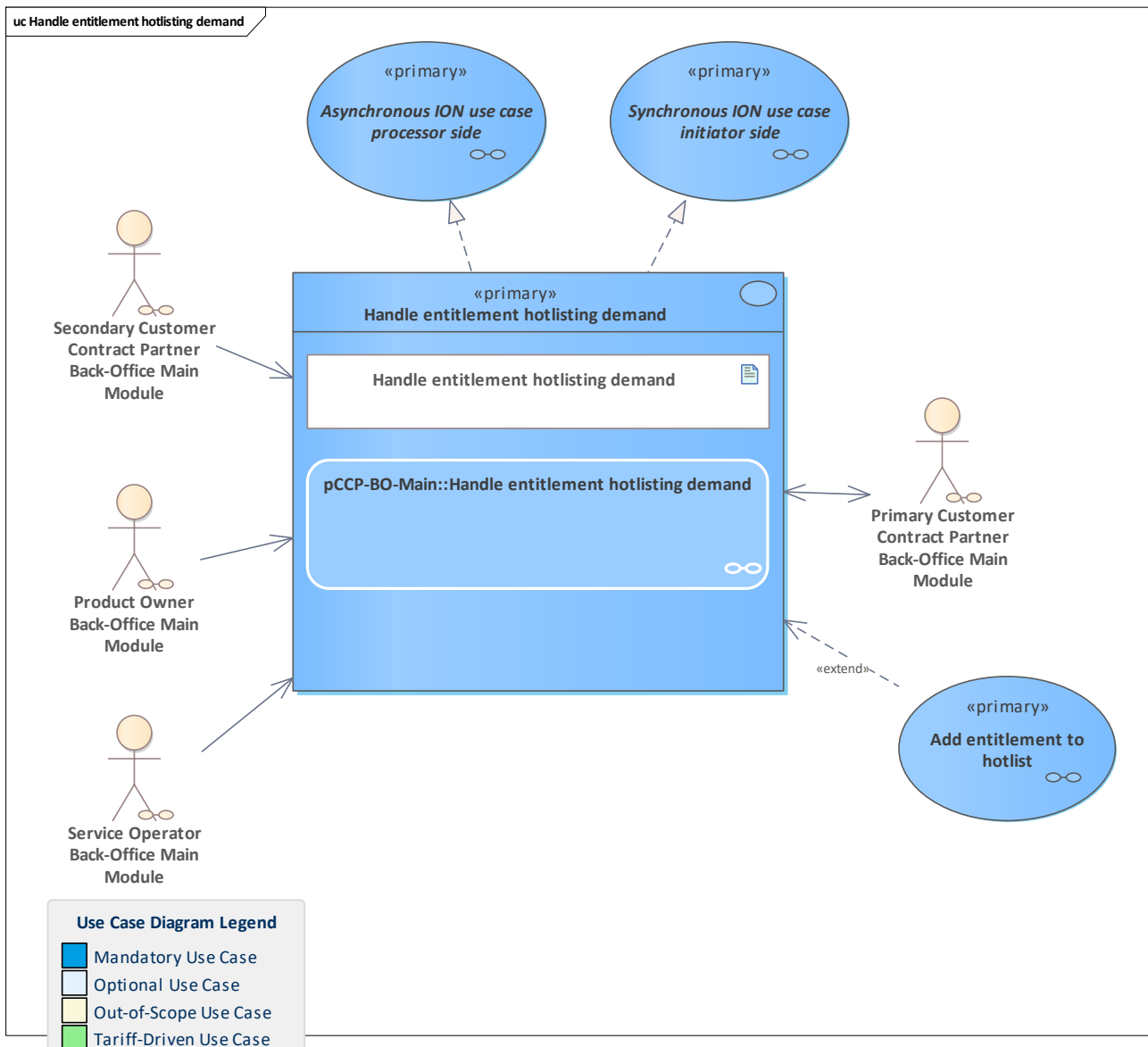


<b>Use Case</b>	<u>Handle revocation for application hotlisting demand</u>
<b>Description</b>	<p>The pCCP receives a revocation request for the application hotlisting demand.</p> <p>The pCCP checks the requests and its information in its application management.</p> <p>If the result is positive and there is no current hotlisting reason, the application will be removed from the hotlist.</p> <p><b>Note:</b> if the pCCP is a new application owner of an existing application instance, the pCCP must have previously transferred the information about old hotlisting demands (if any), especially the ION message references.</p>
<b>Initiating Actor</b>	<u>Service Operator Back-Office Main Module</u> <u>Secondary Customer Contract Partner Back-Office Main Module</u>



	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Remove application from hotlist</a>
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a> / <a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Revoke application hotlisting demand</a> : <a href="#">revokeApplicationHotlistingDemand</a>
<b>Outputs</b>	<a href="#">Revoke application hotlisting demand response</a> : <a href="#">revokeApplicationHotlistingDemandResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO DUPLICATE HOTLISTING DEMAND REVOCATION</a> <a href="#">E CO NO MATCHING HOTLISTING DEMAND TO REVOKE</a> <a href="#">Hotlisting revocation rejected</a> : <a href="#">revokeApplicationHotlistingDemandException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle revocation for application hotlisting demand</a>

## 9.2.3 Handle entitlement hotlisting demand

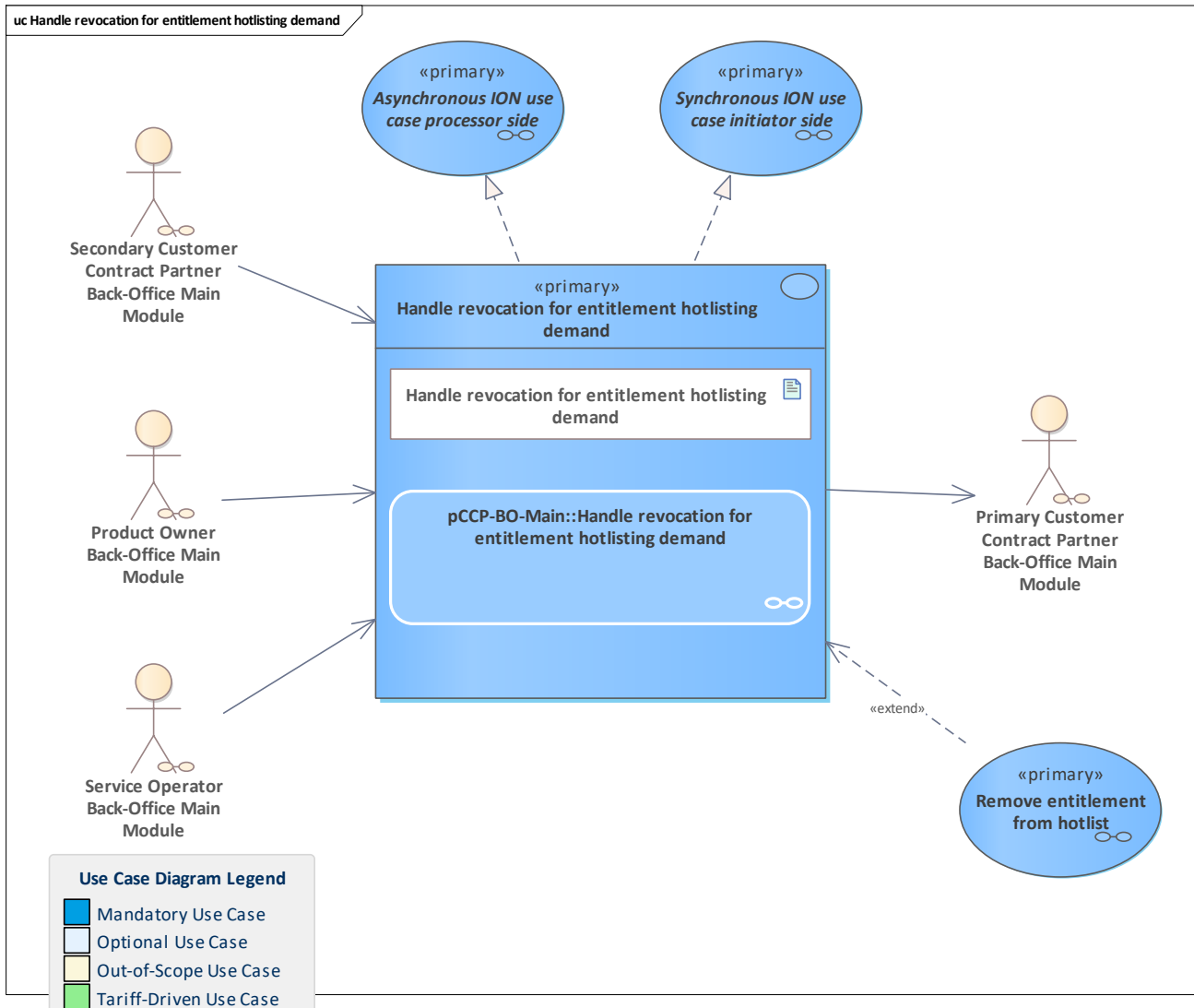


Use Case	Handle entitlement hotlisting demand
Description	<p>The pCCP that issued the entitlement to a customer checks the hotlisting demand.</p> <p>If the result of the check requires a hotlisting, the hotlist service system will be called to add the entitlement to the hotlist.</p> <p>Otherwise, the demand will be rejected and there is no need to communicate with the hotlist service system. The original caller of the demand is informed.</p> <p>Please note that if there is more than one demand for hotlisting the same entitlement, this has to be considered in the check for a required hotlisting but will not result in an exception. Especially for the monitoring of a third-party system, it must be possible to demand hotlisting even if the same object had previously been demanded for hotlisting.</p> <p>Several hotlisting requests for an entitlement can be received. The</p>



	<p>final decision for a hotlisting must be decided in the pCCP, by considering all hotlisting demands, hotlisting demand revocations and hotlisting orders.</p> <p>If the entitlement is already hotlisted, the demand will not cause a further hotlist request towards the hotlist service system.</p> <p><b>Note:</b> in the ION context, the use case is asynchronous as processor (process the demand, <a href="#">Asynchronous ION use case processor side</a>) and a synchronous use case as initiator (<a href="#">Synchronous ION use case initiator side</a>) due to the request to the hotlist service for adding the entitlement to the hotlist.</p>
<b>Initiating Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a> <a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Add entitlement to hotlist</a>
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a> / <a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Demand entitlement hotlisting : demandEntitlementHotlisting</a>
<b>Outputs</b>	<a href="#">Demand entitlement hotlisting response : demandEntitlementHotlistingResponse</a>
<b>Error Cases</b>	<a href="#">E_CO_UNKNOWN_BLOCKING_REASON</a> <a href="#">E_CO_UNKNOWN_BLOCKING_MODE</a> <a href="#">E_CO_WRONG_SECURITY_LEVEL</a> <a href="#">E_CCP_RECEIVER_IS_NOT_ENTITY_OWNER</a> <a href="#">E_CCP_PRODUCT_ID_NOT_MATCHING_DATABASE</a> <a href="#">E_CCP_APP_INSTANCE_ID_NOT_MATCHING_DATABASE</a> <a href="#">E_CCP_ENTITY_ALREADY_TERMINATED</a> <a href="#">Demand entitlement hotlisting exception : demandEntitlementHotlistingException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle entitlement hotlisting demand</a>

## 9.2.4 Handle revocation for entitlement hotlisting demand



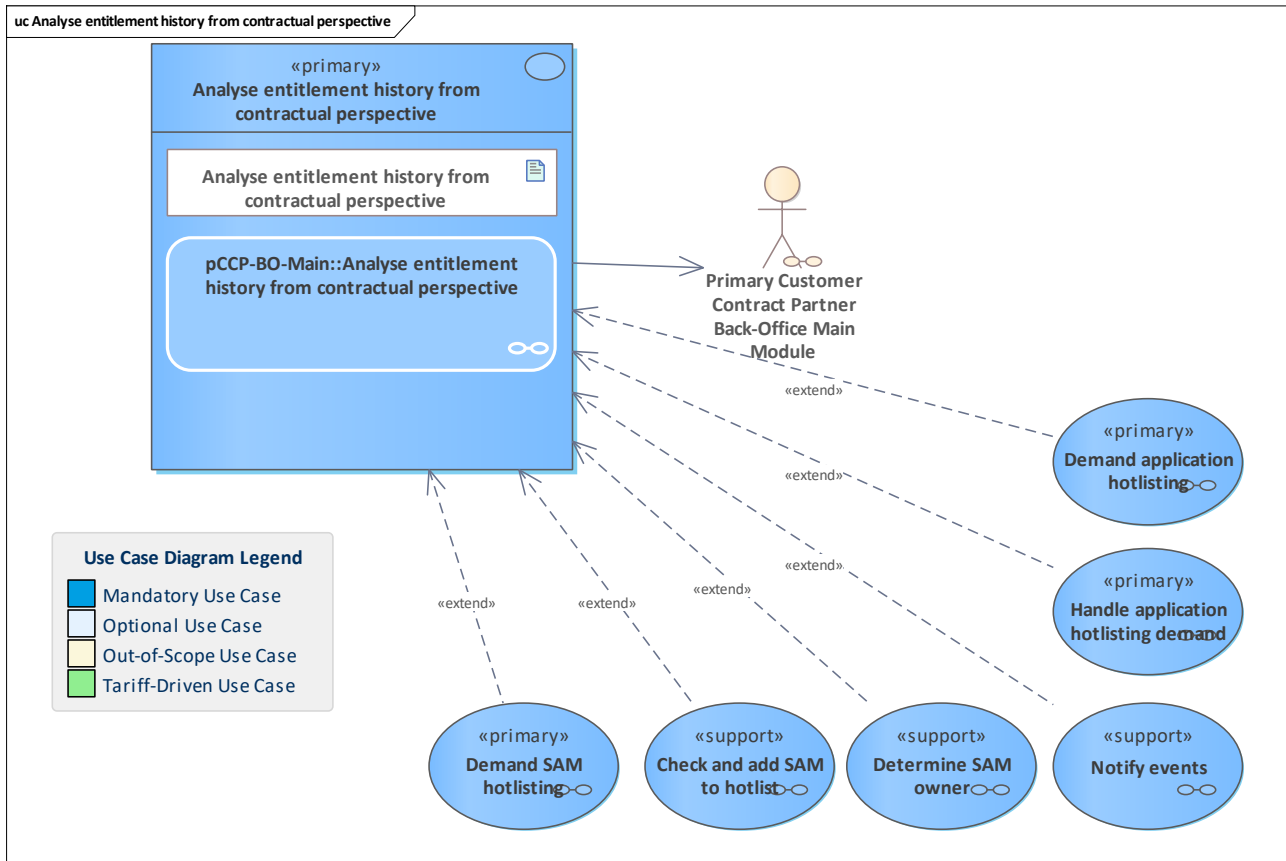
<b>Use Case</b>	<a href="#">Handle revocation for entitlement hotlisting demand</a>
<b>Description</b>	<p>The pCCP receives a revocation request for the entitlement hotlisting demand.</p> <p>The pCCP checks the requests and its information in its entitlement management.</p> <p>If the result is positive and there is no current hotlisting reason, the entitlement will be removed from the hotlist.</p>
<b>Initiating Actor</b>	<a href="#">Secondary Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Remove entitlement from hotlist</a>
<b>Linked Use Cases</b>	





<b>(Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Revoke entitlement hotlisting demand :</a> <a href="#">revokeEntitlementHotlistingDemand</a>
<b>Outputs</b>	<a href="#">Revoke entitlement hotlisting demand response :</a> <a href="#">revokeEntitlementHotlistingDemandResponse</a>
<b>Error Cases</b>	<a href="#">E CO DUPLICATE HOTLISTING DEMAND REVOCATION</a> <a href="#">E CO NO MATCHING HOTLISTING DEMAND TO REVOKE</a> <a href="#">Hotlisting revocation rejected :</a> <a href="#">revokeEntitlementHotlistingDemandException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle revocation for entitlement hotlisting demand</a>

## 9.2.5 Analyse entitlement history from contractual perspective

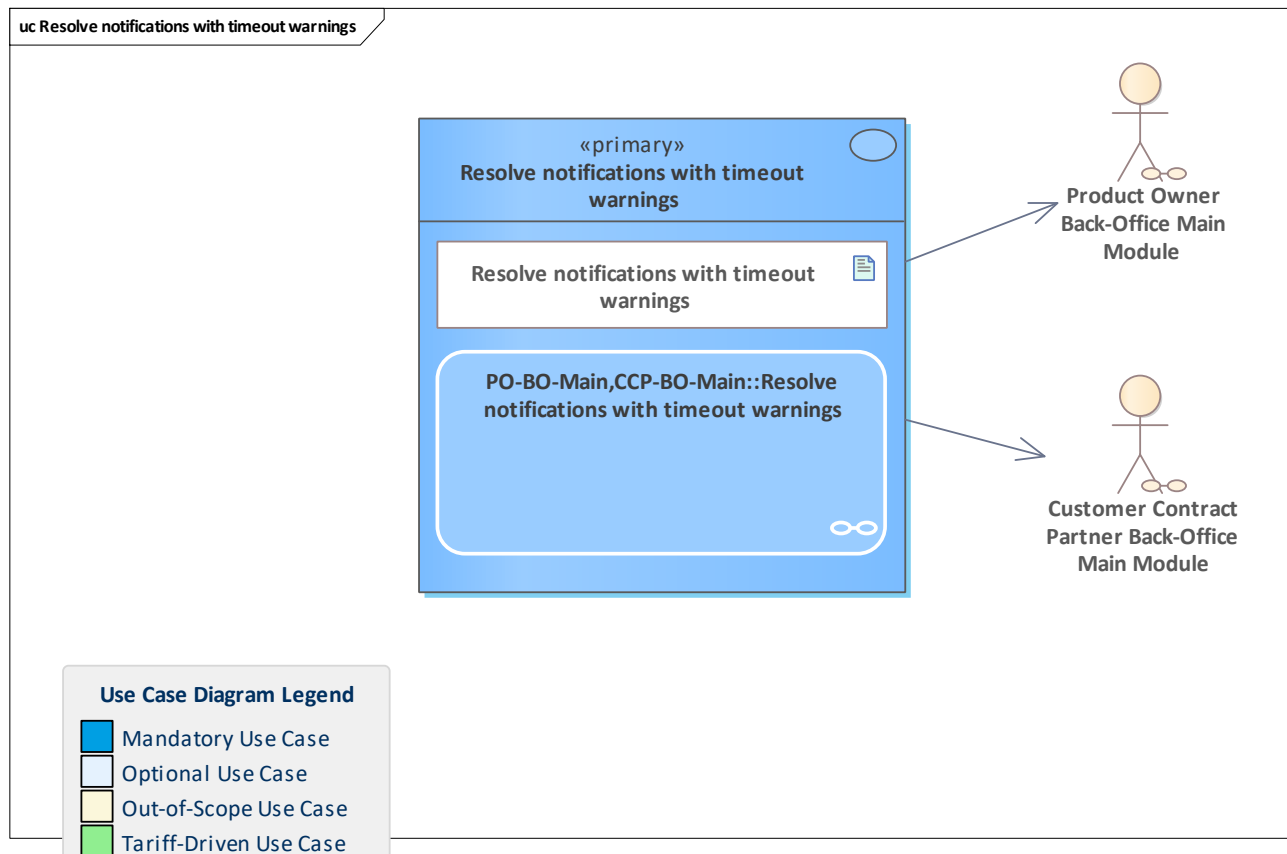


<b>Use Case</b>	<a href="#">Analyse entitlement history from contractual perspective</a>
<b>Description</b>	Analyse the entitlement history for inconsistencies or gaps in the usage history. PO and pCCP both need to have all information about their entitlements, which means they need to have all notifications regarding them.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Handle application hotlisting demand</a> / <a href="#">Notify events</a> / <a href="#">Determine SAM owner</a> / <a href="#">Check and add SAM to hotlist</a> / <a href="#">Demand SAM hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle application hotlisting demand</a> / <a href="#">Determine SAM owner</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	



<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Analyse entitlement history from contractual perspective</a>
-------------------------	----------------------------------------------------------------------------------------

## 9.2.6 Resolve notifications with timeout warnings



<b>Use Case</b>	<u>Resolve notifications with timeout warnings</u>
<b>Description</b>	The system periodically tries to resolve notifications with timeout warnings. A timeout warning can be resolved positively (determining that the action permanently changed the user medium) or negatively (determining the user medium performed a roll-back and was unchanged). When a notification has its timeout warning positively resolved, it can enter regular system processing.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Product Owner Back-Office Main Module</u> <u>Customer Contract Partner Back-Office Main Module</u>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	



<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">PO-BO-Main,CCP-BO-Main::Resolve notifications with timeout warnings</a>

# 10 Basic Bundle CCP-System - UM with Application

Functionality bundle that covers all the basic use cases required for the CCP back-office system when employing user media with an application (e.g. chip cards).

## 10.1 Overview

Handle application blocked notification from contractual perspective

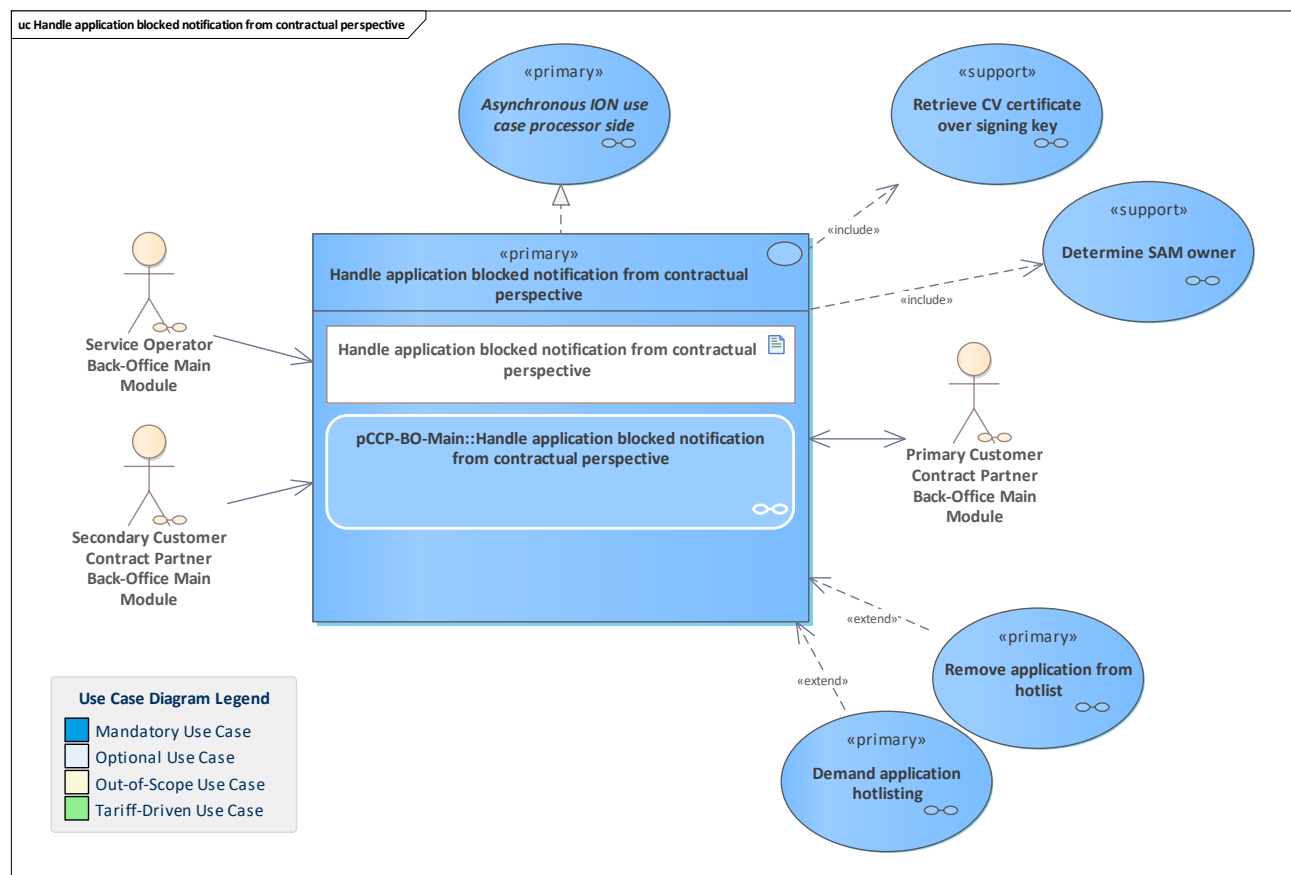
Handle entitlement blocked notification from contractual perspective

Analyse application history from contractual perspective

Check entitlement notifications against issuance notification from contractual perspective

## 10.2 Use Cases

### 10.2.1 Handle application blocked notification from contractual perspective

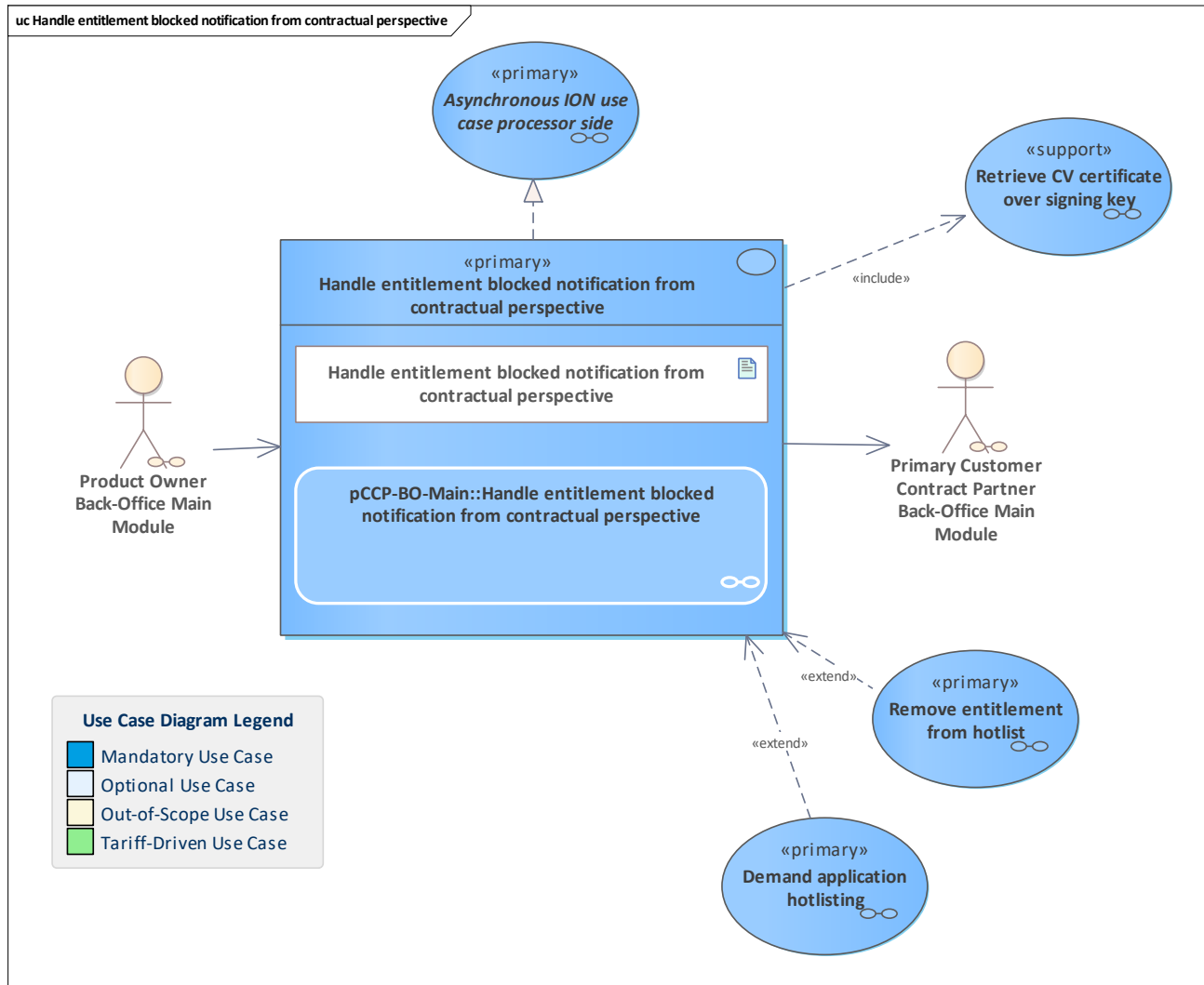


<b>Use Case</b>	<u>Handle application blocked notification from contractual perspective</u>
-----------------	-----------------------------------------------------------------------------



<b>Description</b>	<p>The notification regarding user medium application blocked is received by the pCCP.</p> <p>The pCCP does its checks and monitoring from the contractual perspective regarding the correct execution of blocking. In this context, the SAM owner of the SAM that performed the blocking is determined. Furthermore, the signature of the blocking attestation is verified.</p> <p>If the blocking is correct, the pCCP initiates the removal of the application from the hotlist.</p> <p><b>Note:</b> due to monitoring checks, the application may be demanded to be hotlisted. In this case, the pCCP will add the application directly to the hotlist again (or will not remove it).</p>
<b>Initiating Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a> <a href="#">Secondary Customer Contract Partner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Remove application from hotlist</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Determine SAM owner</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notification of application blocked : notifyApplicationBlocked</a>
<b>Outputs</b>	<a href="#">Notification of application blocked response : notifyApplicationBlockedResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Notify application blocked exception : notifyApplicationBlockedException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle application blocked notification from contractual perspective</a>

## 10.2.2 Handle entitlement blocked notification from contractual perspective



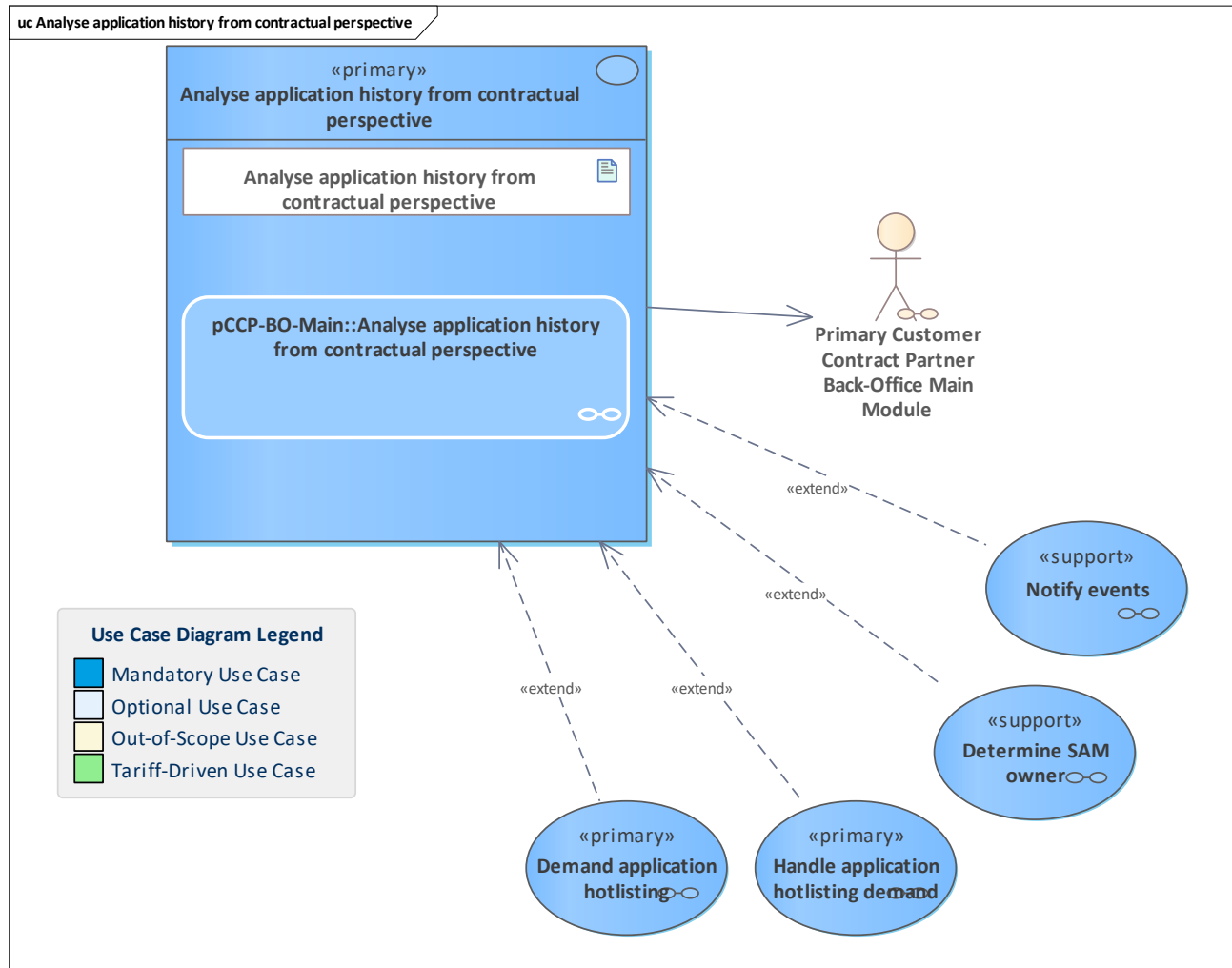
<b>Use Case</b>	<a href="#">Handle entitlement blocked notification from contractual perspective</a>
<b>Description</b>	<p>The entitlement blocked notification is received by the pCCP. The pCCP does its checks and monitoring from the contractual perspective regarding the correct execution of blocking. In this context, the signature of the blocking attestation is verified. If the blocking was correct, the pCCP demands that the entitlement is removed from the hotlist.</p> <p><b>Note:</b> if the pCCP itself performed the blocking, this use case takes inside the use case <a href="#">Handle entitlement blocked notification from operational perspective</a> and is not called by the PO. In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a>.</p> <p><b>Note:</b> due to monitoring checks, the application may be demanded to be hotlisted. In this case, the pCCP will add the application</p>





	directly to the hotlist.
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Remove entitlement from hotlist</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward entitlement blocked notification :</a> <a href="#">forwardEntitlementBlockedNotification</a>
<b>Outputs</b>	<a href="#">Forward entitlement blocked notification response :</a> <a href="#">forwardEntitlementBlockedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward entitlement blocked notification exception :</a> <a href="#">forwardEntitlementBlockedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle entitlement blocked notification from contractual perspective</a>

## 10.2.3 Analyse application history from contractual perspective

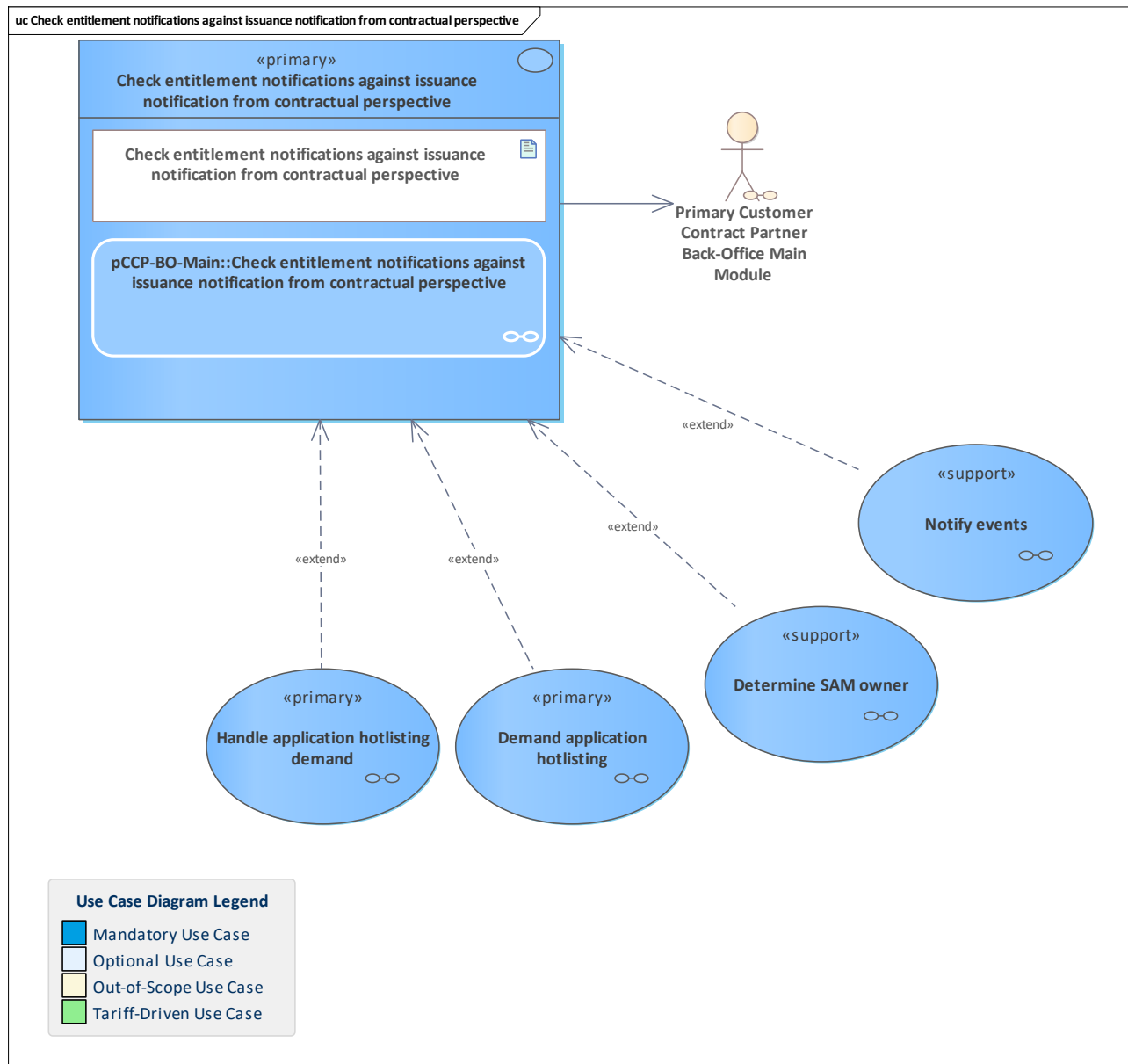


<b>Use Case</b>	<a href="#">Analyse application history from contractual perspective</a>
<b>Description</b>	Analyse the application history for inconsistencies or gaps in their history. The pCCP needs to have all information about their applications, which means they need to have all notifications regarding them. Every issue found shall only be reported once.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Notify events</a> / <a href="#">Determine SAM owner</a> / <a href="#">Handle application hotlisting demand</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle application hotlisting demand</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	



<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Analyse application history from contractual perspective</a>

## 10.2.4 Check entitlement notifications against issuance notification from contractual perspective



<b>Use Case</b>	<a href="#">Check entitlement notifications against issuance notification from contractual perspective</a>
<b>Description</b>	Non-issuing entitlement notifications need to be checked against details that are only contained in the issuance notification, e.g., the entitlement validity period. This also makes sure that every entitlement seen live is known to the system / has been reported as issued.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	<a href="#">Notify events</a> / <a href="#">Demand application hotlisting</a> / <a href="#">Determine SAM</a>



<b>(Extended By)</b>	<a href="#">owner / Handle application hotlisting demand</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle application hotlisting demand</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Check entitlement notifications against issuance notification from contractual perspective</a>



# 11 Basic Bundle CCP-System - UM in Customer Center

This functionality bundle includes use cases that are normally only carried out in the customer centre.

## 11.1 Overview

Handle application unblocked notification from operational perspective

Handle application unblocked notification from contractual perspective

Handle entitlement unblocked notification from operational perspective

Handle entitlement unblocked notification from contractual perspective

Optional: Handle lost user medium with application

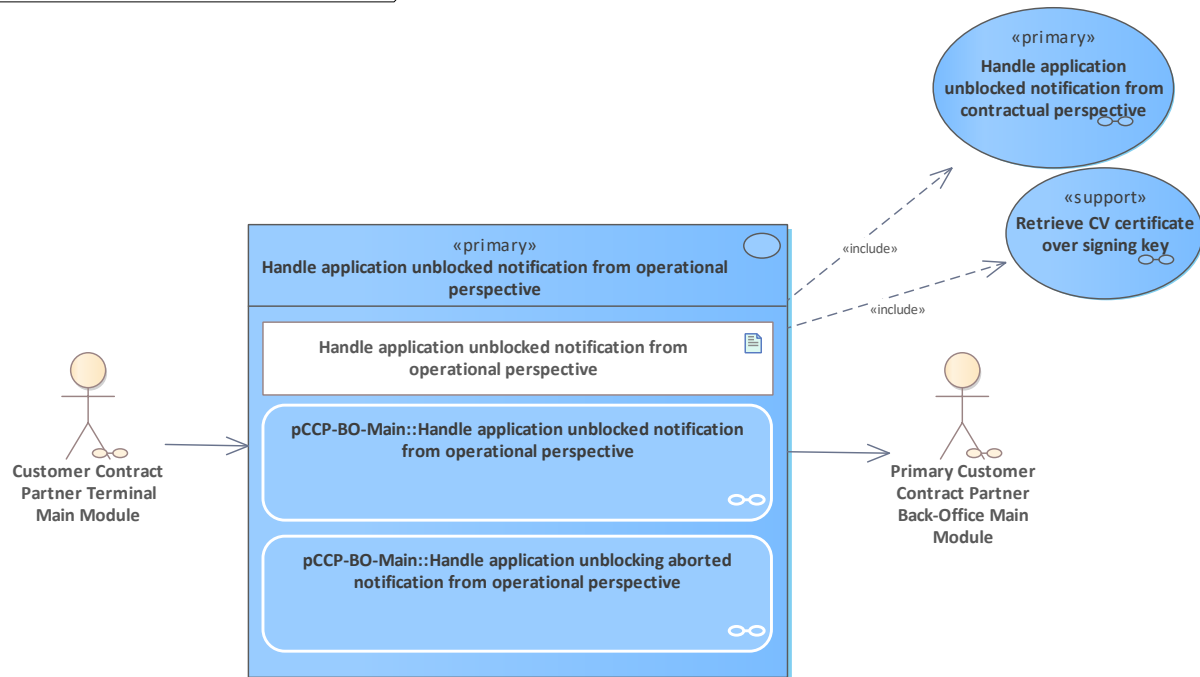
Optional: Handle application terminated notification from operational perspective

Optional: Handle application terminated notification from contractual perspective

## 11.2 Use Cases

### 11.2.1 Handle application unblocked notification from operational perspective

uc Handle application unblocked notification from operational perspective



Use Case Diagram Legend

- Mandatory Use Case
- Optional Use Case
- Out-of-Scope Use Case
- Tariff-Driven Use Case

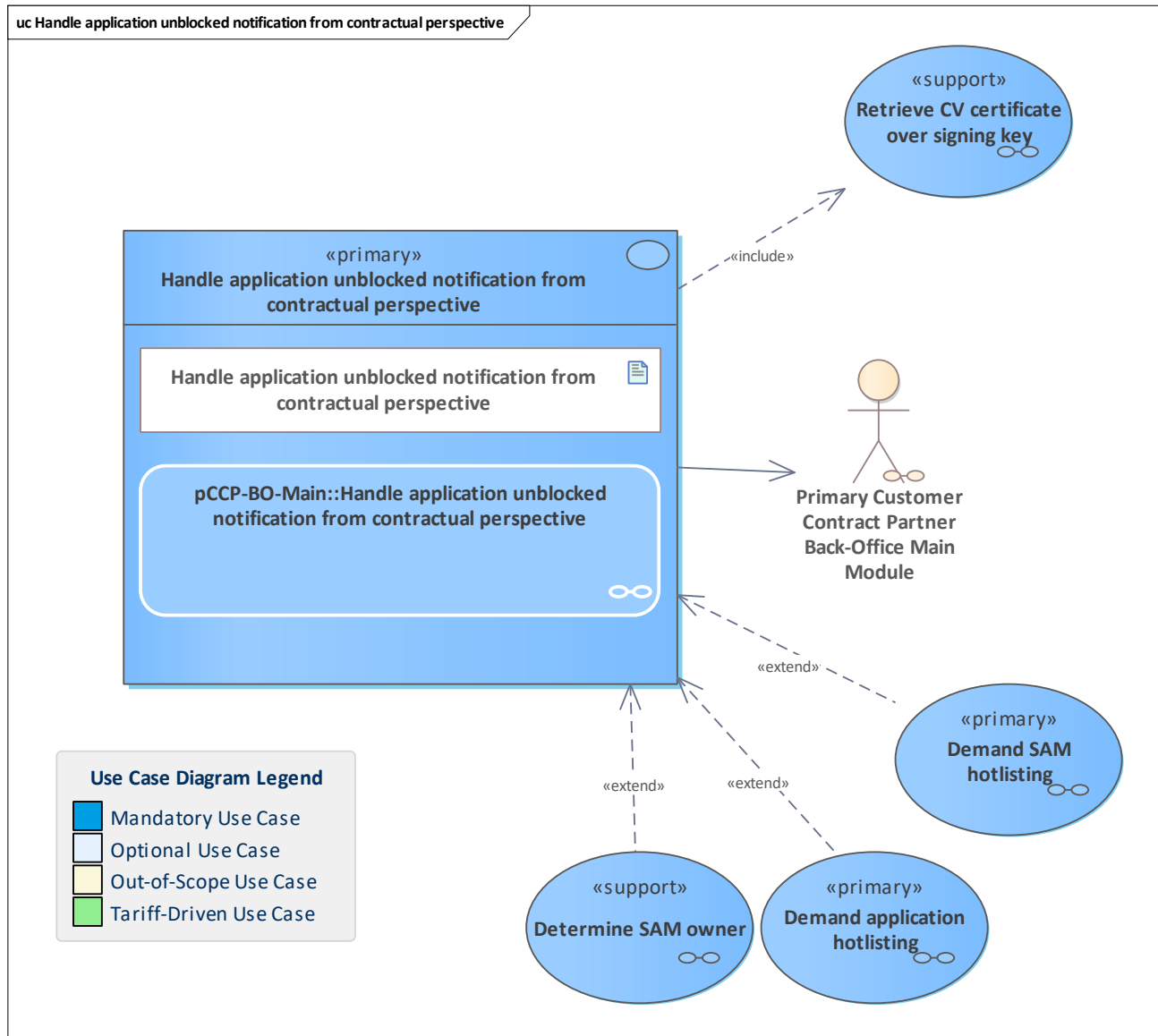
<b>Use Case</b>	<a href="#">Handle application unblocked notification from operational perspective</a>
<b>Description</b>	<p>A terminal has unblocked the application in an user medium with an application.</p> <p>In this use case, the pCCP receives the notification from its terminal and runs certain checks from operational perspective, such as the signature verification of the unblocking attestation. Then the contractual checks are done.</p> <p>As an alternative flow, a potential transaction abortion is handled.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle application unblocked notification from contractual perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notify application unblocked from terminal</a> : <a href="#">tNotifyApplicationUnblocked</a>



<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle application unblocked notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify application unblocking aborted from terminal : tNotifyApplicationUnblockingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle application unblocking aborted notification from operational perspective</a>



## 11.2.2 Handle application unblocked notification from contractual perspective

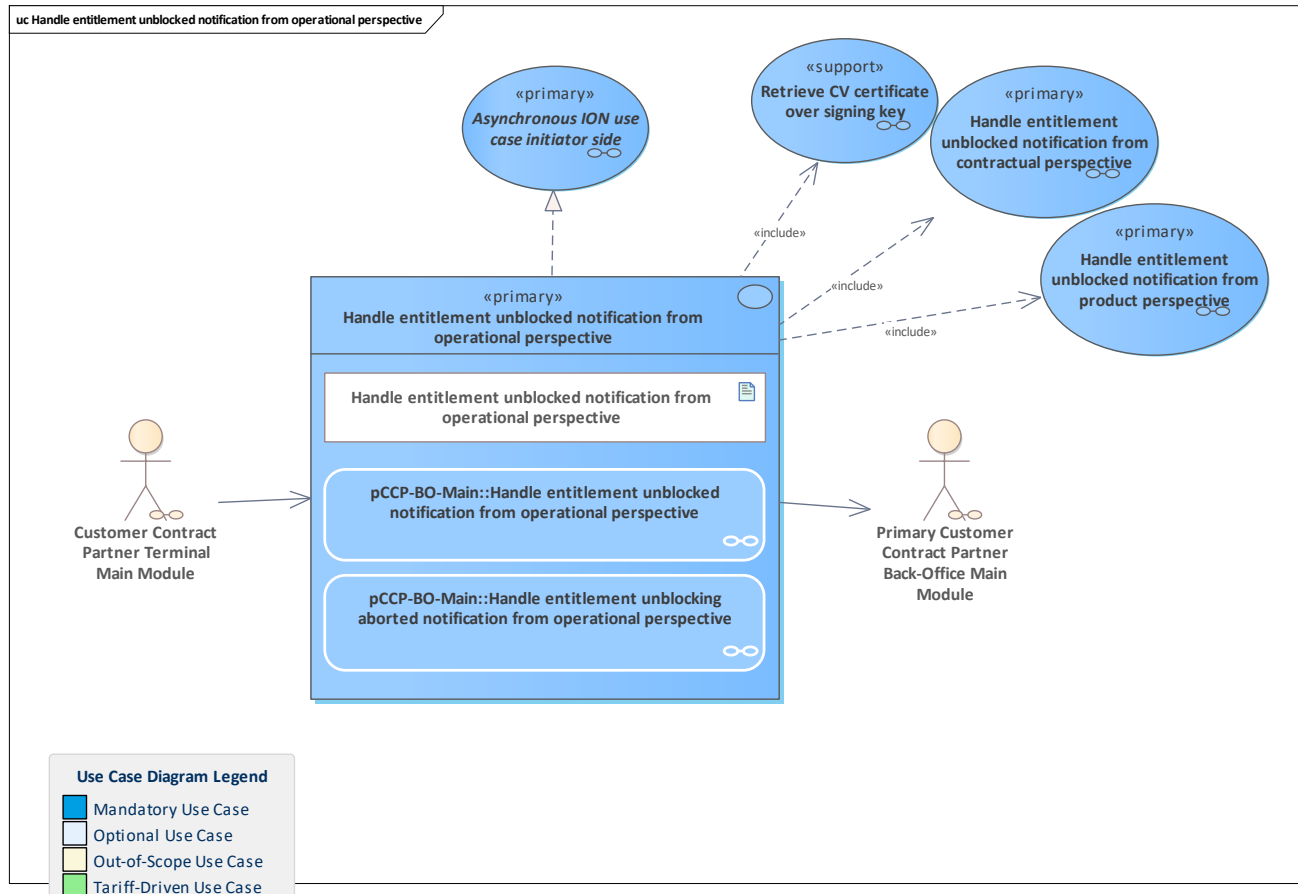


<b>Use Case</b>	<a href="#">Handle application unblocked notification from contractual perspective</a>
<b>Description</b>	The pCCP checks the unblocked UM application notification from the contractual perspective.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Demand SAM hotlisting</a> / <a href="#">Determine SAM owner</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Determine SAM owner</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases</b>	



<b>(Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle application unblocked notification from contractual perspective</a>

## 11.2.3 Handle entitlement unblocked notification from operational perspective

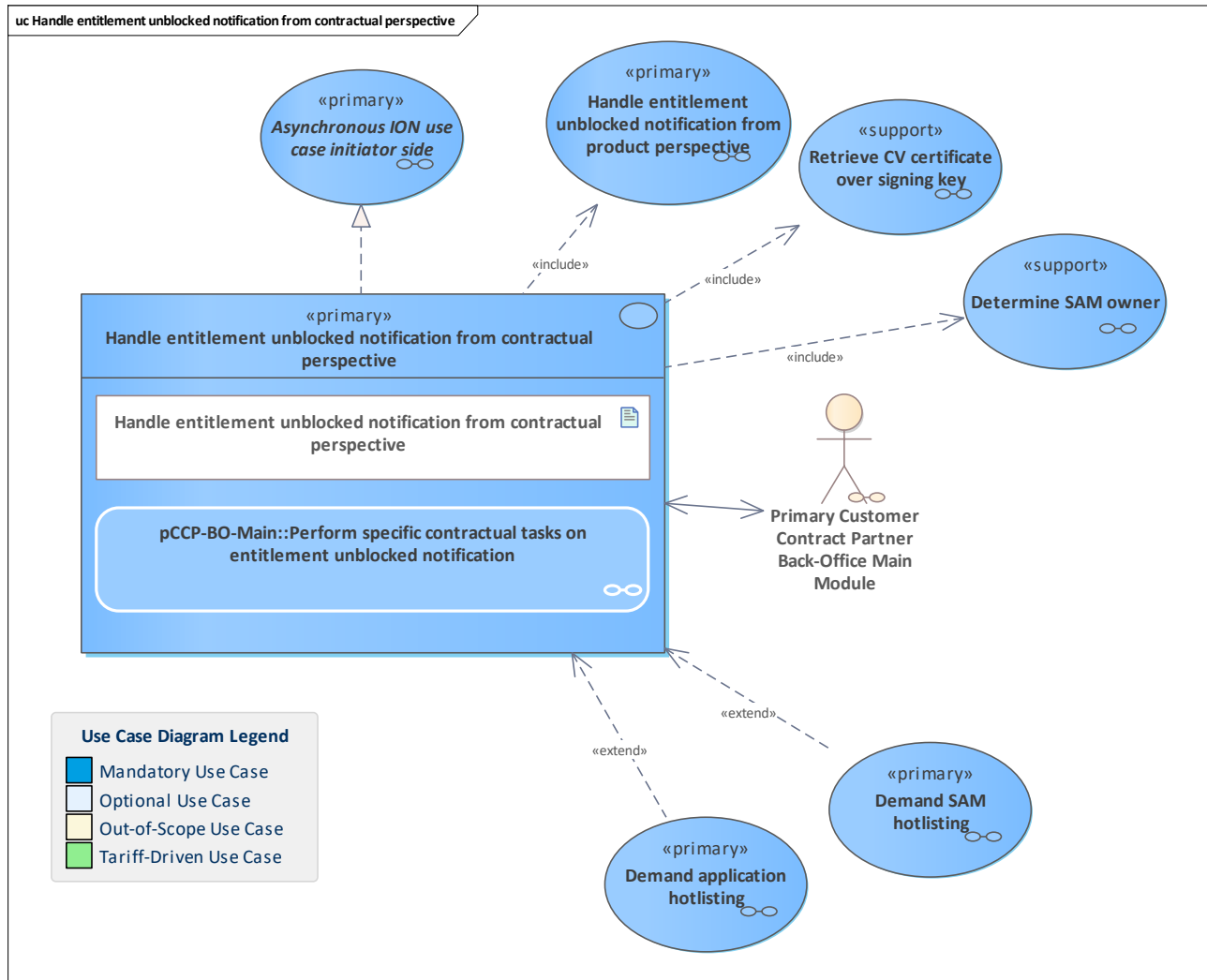


<b>Use Case</b>	<a href="#">Handle entitlement unblocked notification from operational perspective</a>
<b>Description</b>	<p>A terminal has unblocked the entitlement in a user medium with an application.</p> <p>In this use case, the pCCP back-office system receives the notification from the terminal and runs certain checks from the operational perspective such as the signature verification of the attestation of the unblocking. Then the contractual checks are done.</p> <p>Finally, the notification is forwarded to the PO.</p> <p>As an alternative flow, a potential transaction abortion is handled.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement unblocked notification from contractual perspective</a> / <a href="#">Handle entitlement unblocked notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Handle entitlement</a>



	<a href="#">unblocked notification from product perspective / Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notify entitlement unblocked from terminal : tNotifyEntitlementUnblocked</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle entitlement unblocked notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify entitlement unblocking aborted from terminal : tNotifyEntitlementUnblockingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle entitlement unblocking aborted notification from operational perspective</a>

## 11.2.4 Handle entitlement unblocked notification from contractual perspective

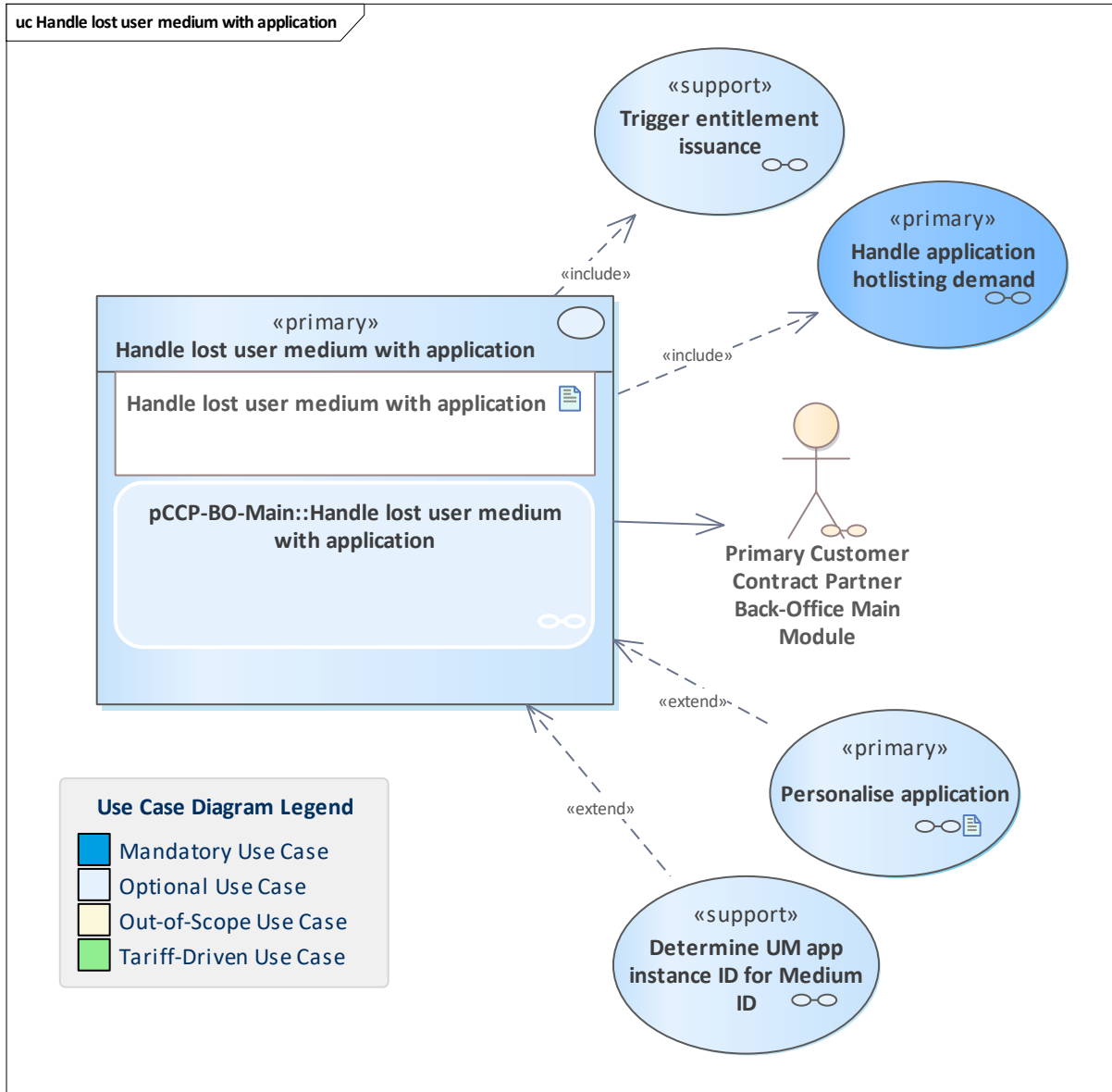


<b>Use Case</b>	<a href="#">Handle entitlement unblocked notification from contractual perspective</a>
<b>Description</b>	Handle an entitlement unblocked notification from the contractual perspective. The pCCP does its checks and monitoring from the contractual perspective regarding the correct execution of the unblock action. In this context, the signature of the embedded attestation is verified.
<b>Initiating Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Demand SAM hotlisting</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement unblocked notification from product perspective</a> / <a href="#">Determine SAM owner</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Determine SAM owner</a> / <a href="#">Retrieve CV certificate over signing key</a> /



	<a href="#">Determine SAM owner</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Perform specific contractual tasks on entitlement unblocked notification</a>

## 11.2.5 Optional: Handle lost user medium with application



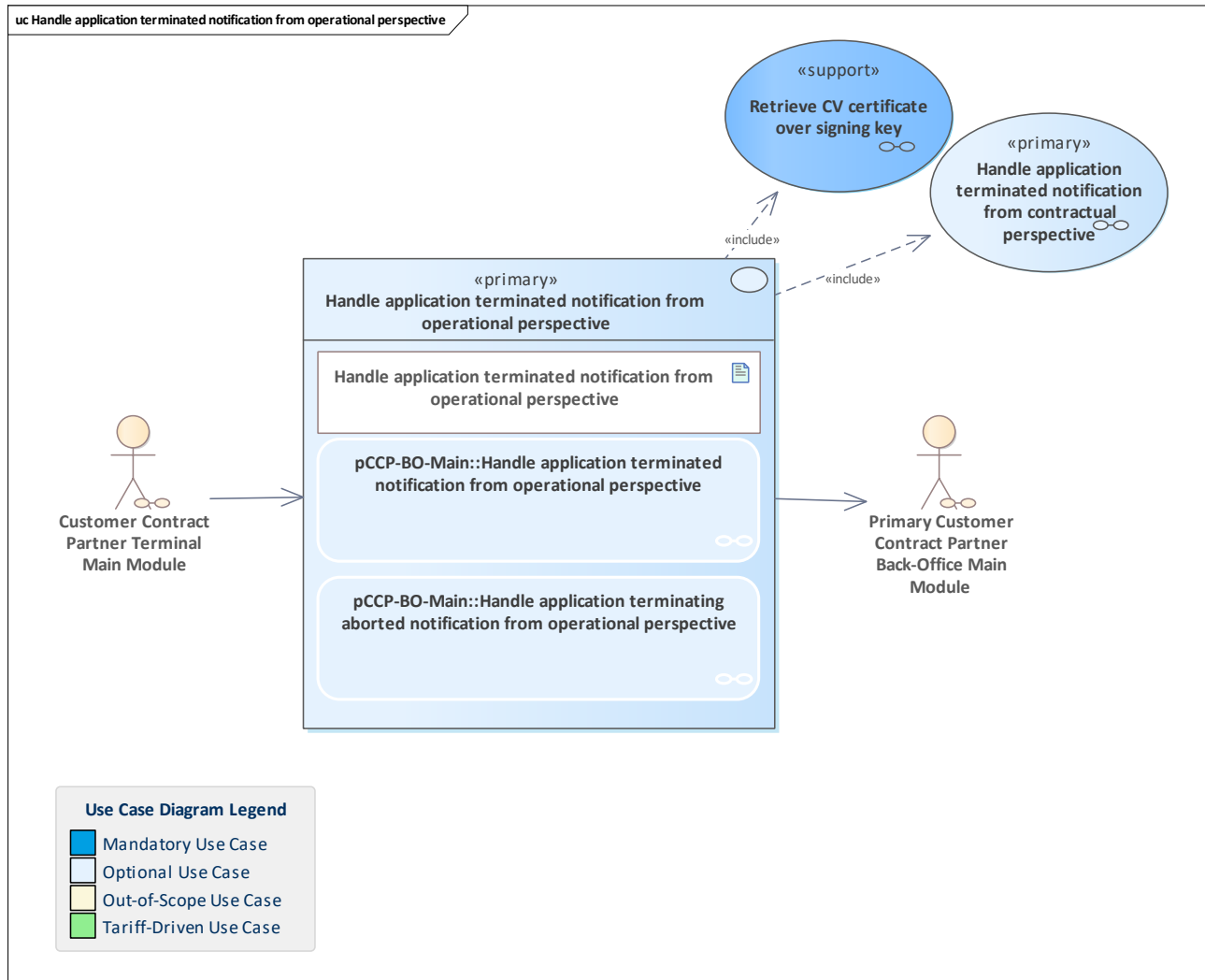
<b>Use Case</b>	<a href="#">Handle lost user medium with application</a>
<b>Description</b>	<p>A user medium with an application is lost. Therefore, the application must be hotlisted and entitlements on the lost user medium can be re-issued on a new user medium with application.</p> <p>Please note that not yet valid entitlements are not reissued.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Determine UM app instance ID for Medium ID</a> / <a href="#">Personalise application</a>
<b>Linked Use Cases</b>	<a href="#">Trigger entitlement issuance</a> / <a href="#">Handle application hotlisting</a>



<b>(Includes)</b>	<a href="#">demand</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">pCCP : OrganisationId</a> <a href="#">Valid on : ZonedDate</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle lost user medium with application</a>



## 11.2.6 Optional: Handle application terminated notification from operational perspective

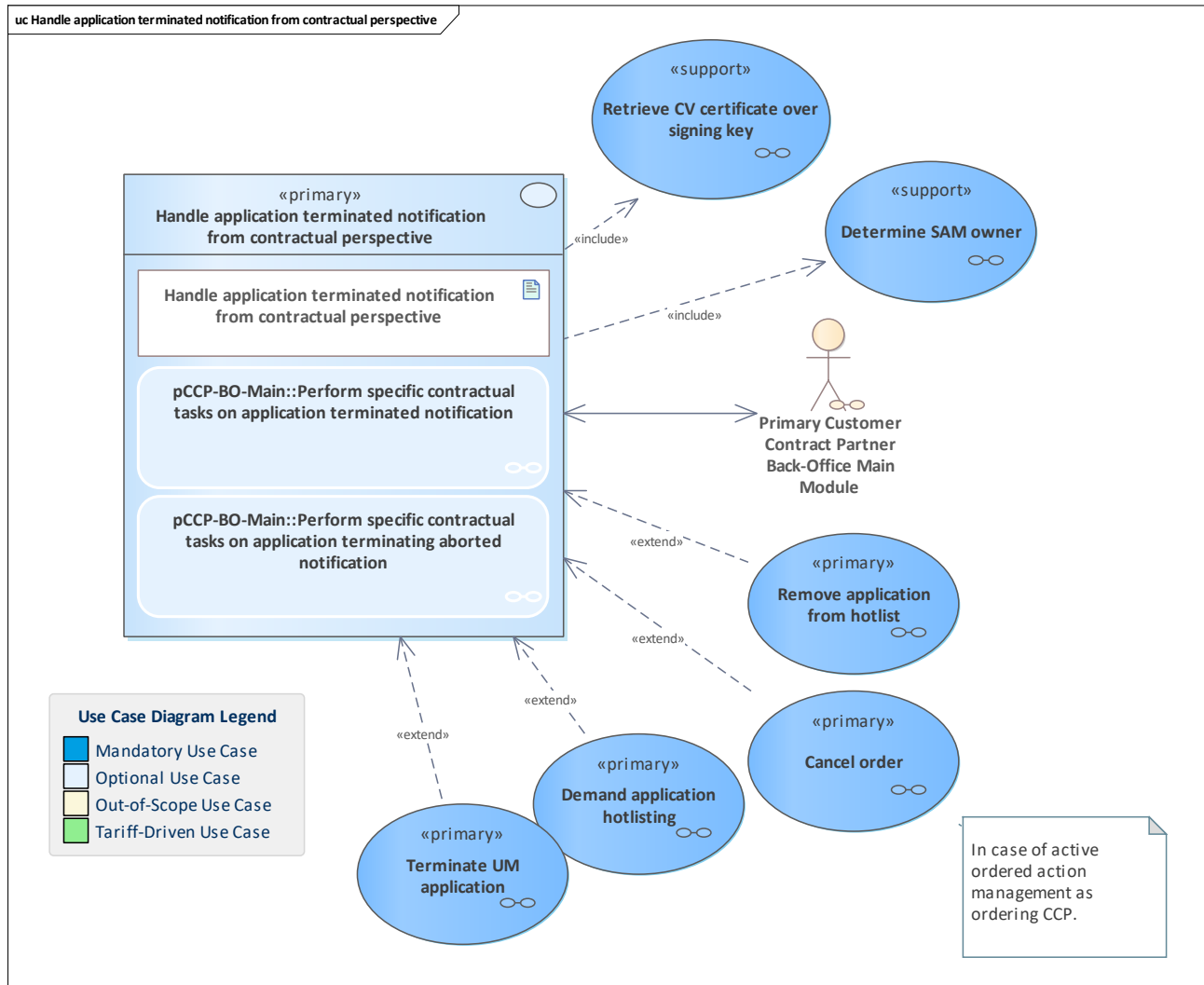


<b>Use Case</b>	<a href="#">Handle application terminated notification from operational perspective</a>
<b>Description</b>	Handle an application termination from an operational perspective. The notification is sent from the CCP terminal to the pCCP back-office system. The pCCP does its operational checks and monitoring and triggers the contractual handling. If the termination transaction was aborted, the notification is registered and SAM counter values are stored for consistent monitoring.
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle application terminated notification from contractual perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV</a>



	<a href="#">certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">t Notify Application Terminated: tNotifyApplicationTerminated</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle application terminated notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify application terminating aborted : tNotifyApplicationTerminatingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle application terminating aborted notification from operational perspective</a>

## 11.2.7 Optional: Handle application terminated notification from contractual perspective



<b>Use Case</b>	<a href="#">Handle application terminated notification from contractual perspective</a>
<b>Description</b>	Handle an application terminated notification from a contractual perspective. The pCCP back-office system does its contractual checks and monitoring. If the transaction of the termination was aborted, this might cause a clearing case.
<b>Initiating Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Cancel order</a> / <a href="#">Remove application from hotlist</a> / <a href="#">Terminate UM application</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Determine SAM owner</a>



<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID from terminal action : ProcessInstanceId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Perform specific contractual tasks on application terminated notification</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify application terminating aborted : ActionAbortedData</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Perform specific contractual tasks on application terminating aborted notification</a>

# 12 Basic Bundle CCP-System - UM with Customer Data

This optional functionality bundle includes use cases that manage extended customer data on the user medium in interaction between the back-office system with the terminal.

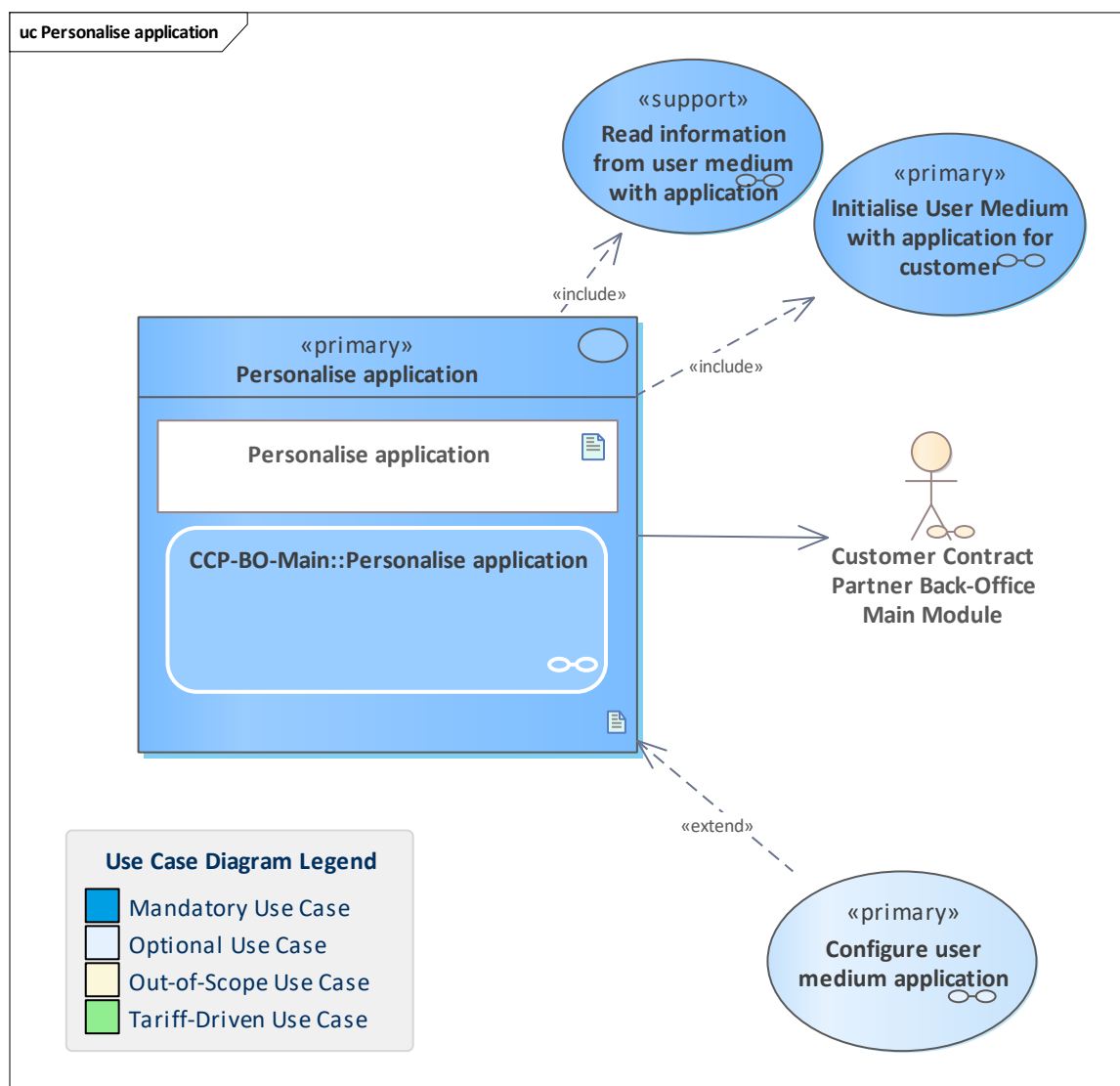
## 12.1 Overview

Personalise application

Process new information about customer and discounts

## 12.2 Use Cases

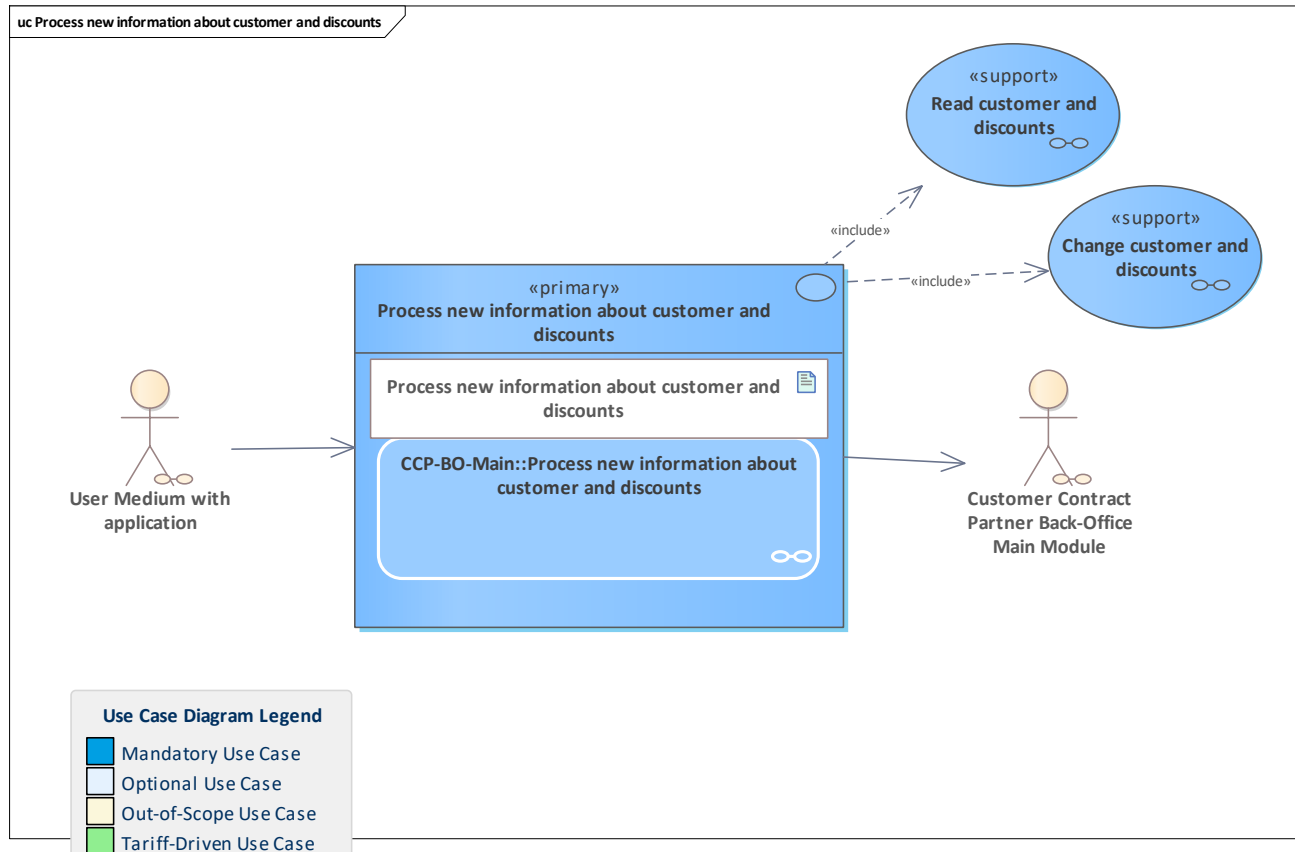
### 12.2.1 Personalise application





<b>Use Case</b>	<a href="#">Personalise application</a>
<b>Description</b>	The CCP back-office system personalises a user medium application instance by first identifying it (using its app instance ID) and then writing the customer-related information to it, which is also stored in the CCP system.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Configure user medium application</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Read information from user medium with application</a> / <a href="#">Initialise User Medium with application for customer</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	<a href="#">UM app instance ID : AppInstanceId</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Personalise application</a>

## 12.2.2 Process new information about customer and discounts



<b>Use Case</b>	<a href="#">Process new information about customer and discounts</a>
<b>Description</b>	The CCP back-office system collects new or changed customer data and triggers the terminal to process this data on the currently involved user medium. Information about the customer and his discounts written on the user medium with application needs to be changed. Therefore, the customer and his discounts are read, changed and written again. Involved entitlements are changed (terminated and issued with the new data) due to the changed customer data.
<b>Initiating Actor</b>	<a href="#">User Medium with application</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a> <a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Read customer and discounts</a> / <a href="#">Change customer and discounts</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	



<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Process new information about customer and discounts</a>



## 13 Basic Bundle CCP-System - Extension for UM with Application

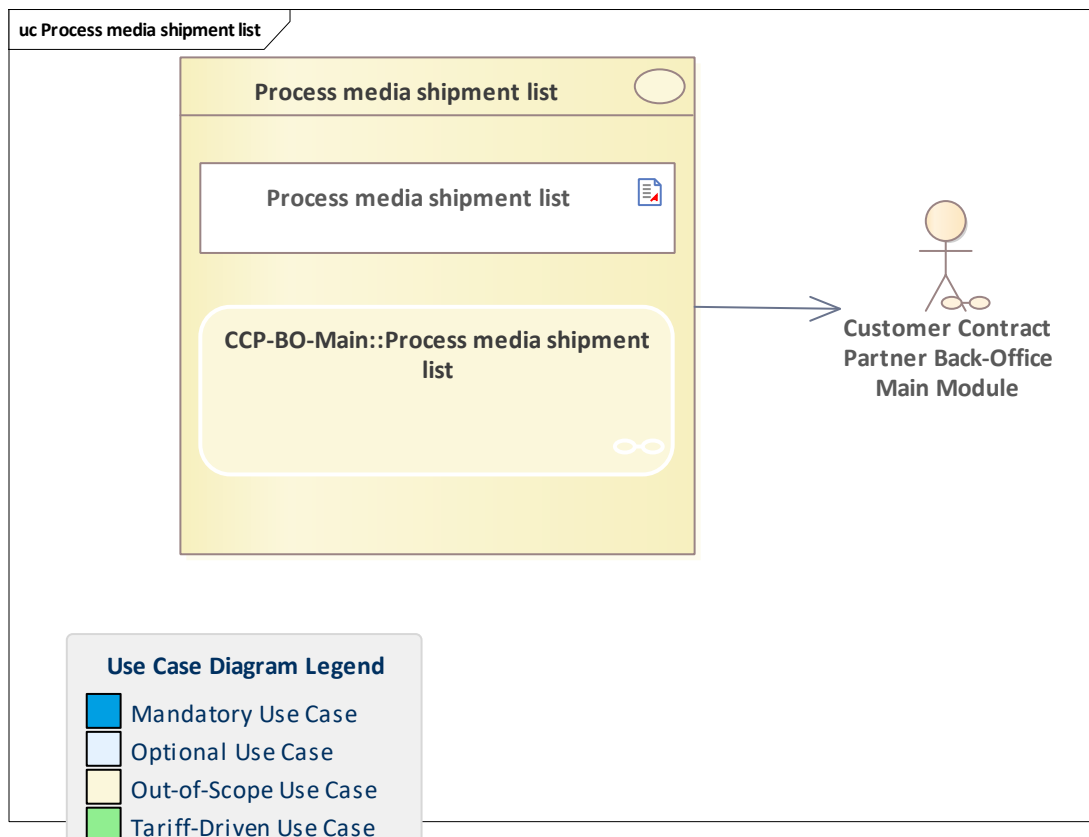
Functionality bundle that covers all the basic use cases required for the CCP back-office system when employing user media with an application (e.g. chip cards) and extended functionality.

### 13.1 Overview

Process media shipment list

### 13.2 Use Cases

#### 13.2.1 Process media shipment list



<b>Use Case</b>	<u>Process media shipment list</u>
<b>Description</b>	The CCP-BO-Main processes the media shipment list for an order of user media. The contained medium IDs and application instance IDs are registered and the order state is changed to <i>received</i> . Apart from the interface, this use case is not specified in detail.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u>
<b>Preconditions</b>	
<b>Postconditions</b>	



<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Media shipment list : mediaShipmentList</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Process media shipment list</a>

# 14 Electronic Ticket Bundle CCP-System

Functionality bundle that covers the use cases for a CCP back-office system with electronic tickets placed on a user medium with application (e.g. chip card).

## 14.1 Overview

## Handle entitlement issued notification from operational perspective

## Handle entitlement issued notification from contractual perspective

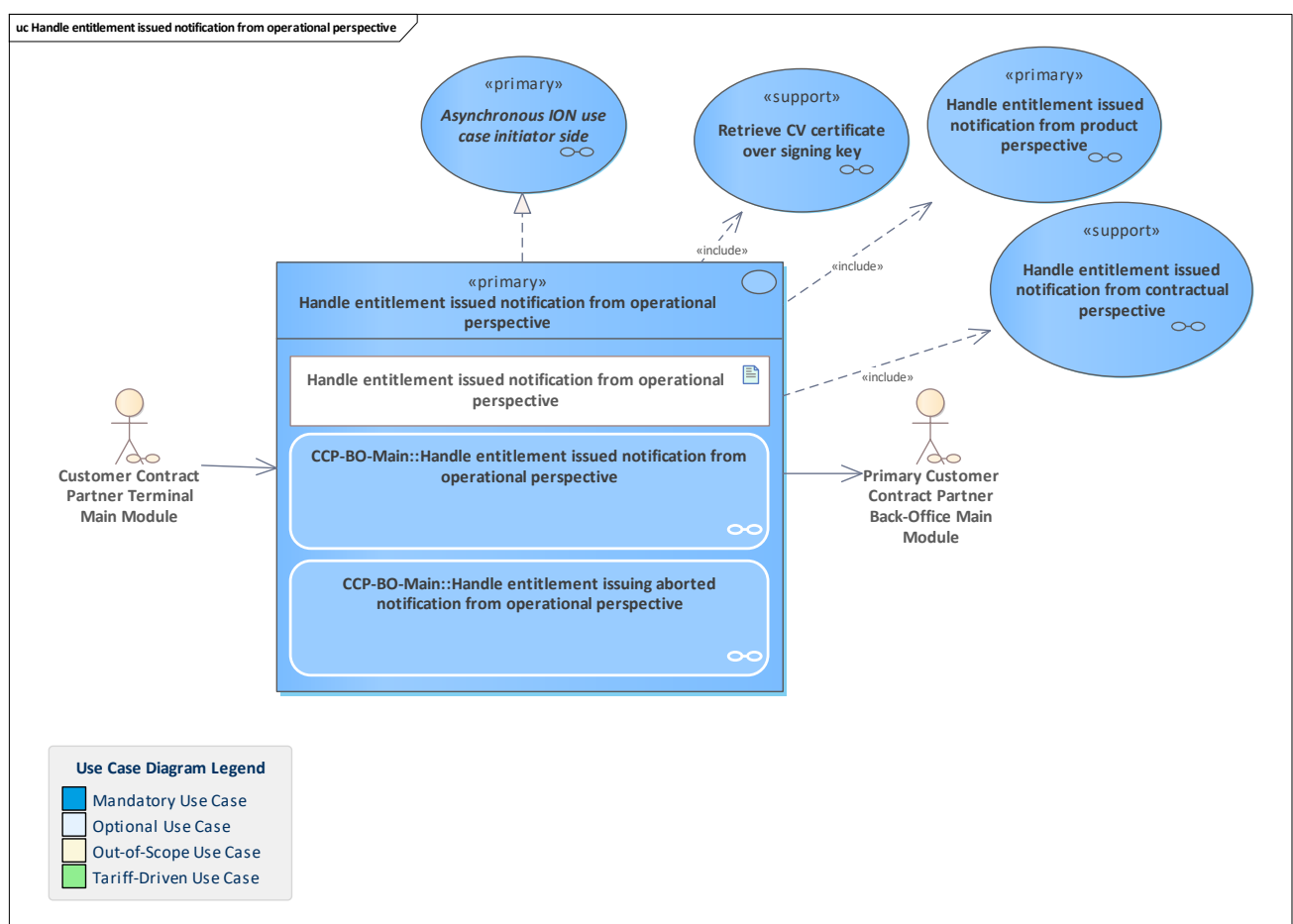
## Handle entitlement terminated notification from operational perspective

## Handle entitlement terminated notification from contractual perspective

## Handle entitlement inspected notification from contractual perspective

## 14.2 Use Cases

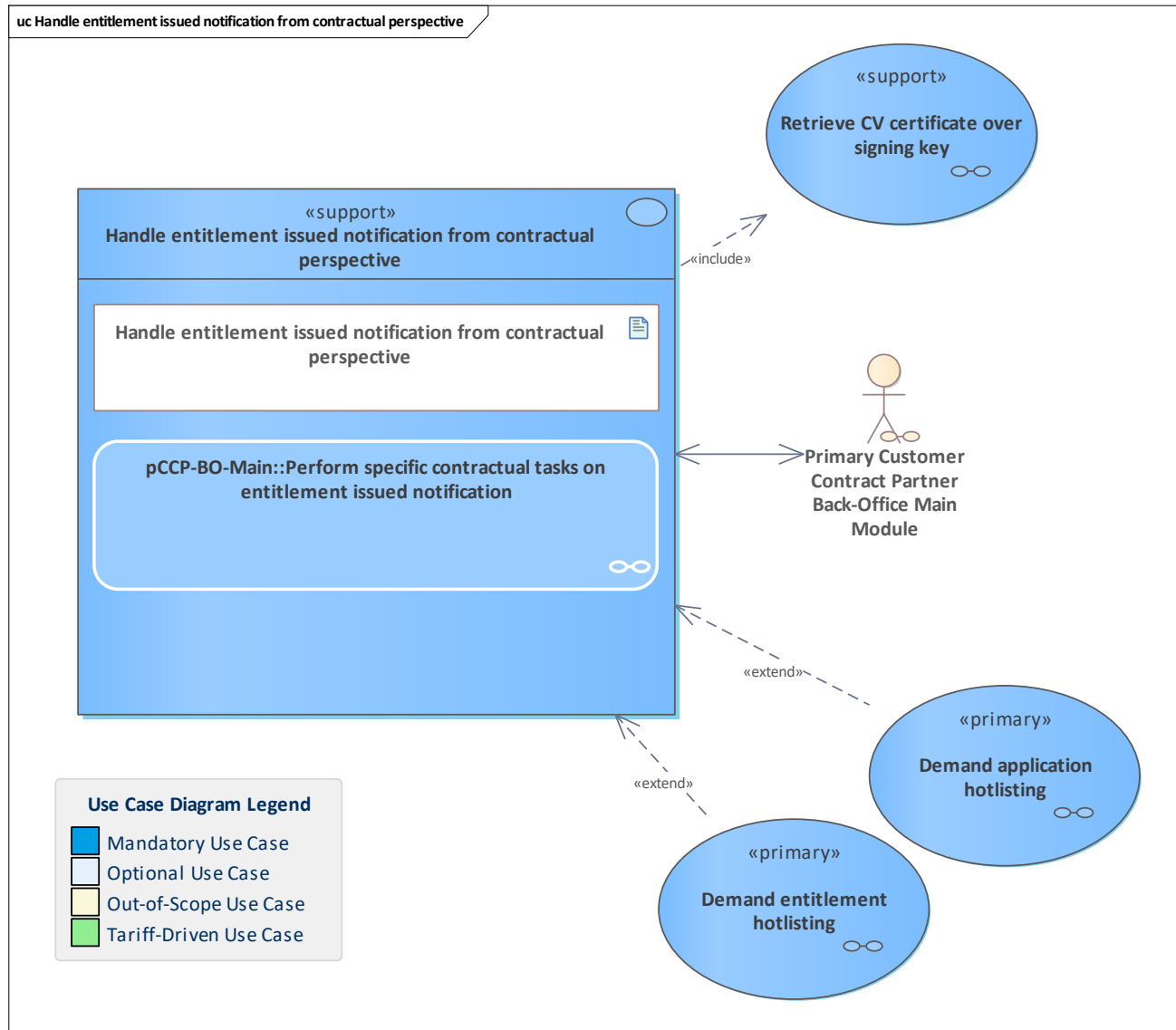
### 14.2.1 Handle entitlement issued notification from operational perspective





<b>Use Case</b>	<a href="#">Handle entitlement issued notification from operational perspective</a>
<b>Description</b>	<p>The pCCP of the entitlement receives the notification about an entitlement issuance and registers it.</p> <p>Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the issuance attestation.</p> <p>Since it is the pCCP, it also does the contractual checks and monitoring.</p> <p>Finally, the notification is forwarded to the responsible PO system. This can be done either with a single message or in a scheduled process with a list of messages.</p> <p>In case of an abortion, the notification is also sent to the PO system to announce the used SAM- and product issuance counters.</p> <p><b>Note:</b> this use case is without the context of ordered action execution.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement issued notification from product perspective</a> / <a href="#">Handle entitlement issued notification from contractual perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify entitlement issuing aborted : tNotifyEntitlementIssuingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement issuing aborted notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify entitlement issued : tNotifyEntitlementIssued</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement issued notification from operational perspective</a>

## 14.2.2 Handle entitlement issued notification from contractual perspective

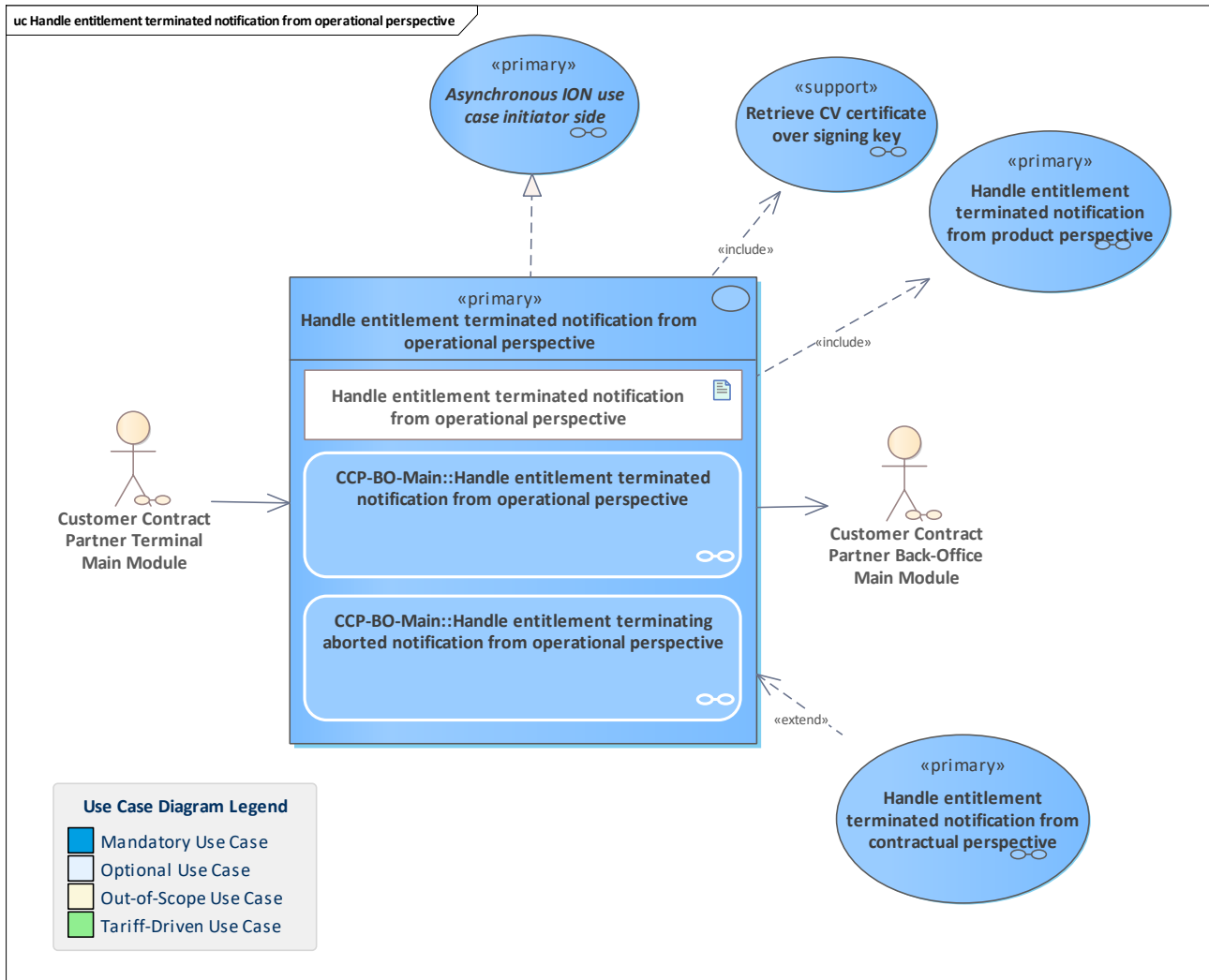


<b>Use Case</b>	<a href="#">Handle entitlement issued notification from contractual perspective</a>
<b>Description</b>	<p>Handle a notification about the issuance of an owned entitlement from the contractual perspective by performing the related checks and monitoring.</p> <p>Note that the application instance ID of the user medium the entitlement was issued to can be uniquely identified via <i>umAppInstanceId</i>, which is part of the <a href="#">SignedEntitlementIssuedAttestation</a> that is contained in the <a href="#">EntitlementIssuedNotification</a>.</p>
<b>Initiating Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	<a href="#">Demand entitlement hotlisting</a> / <a href="#">Demand application hotlisting</a>



<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Perform specific contractual tasks on entitlement issued notification</a>

## 14.2.3 Handle entitlement terminated notification from operational perspective



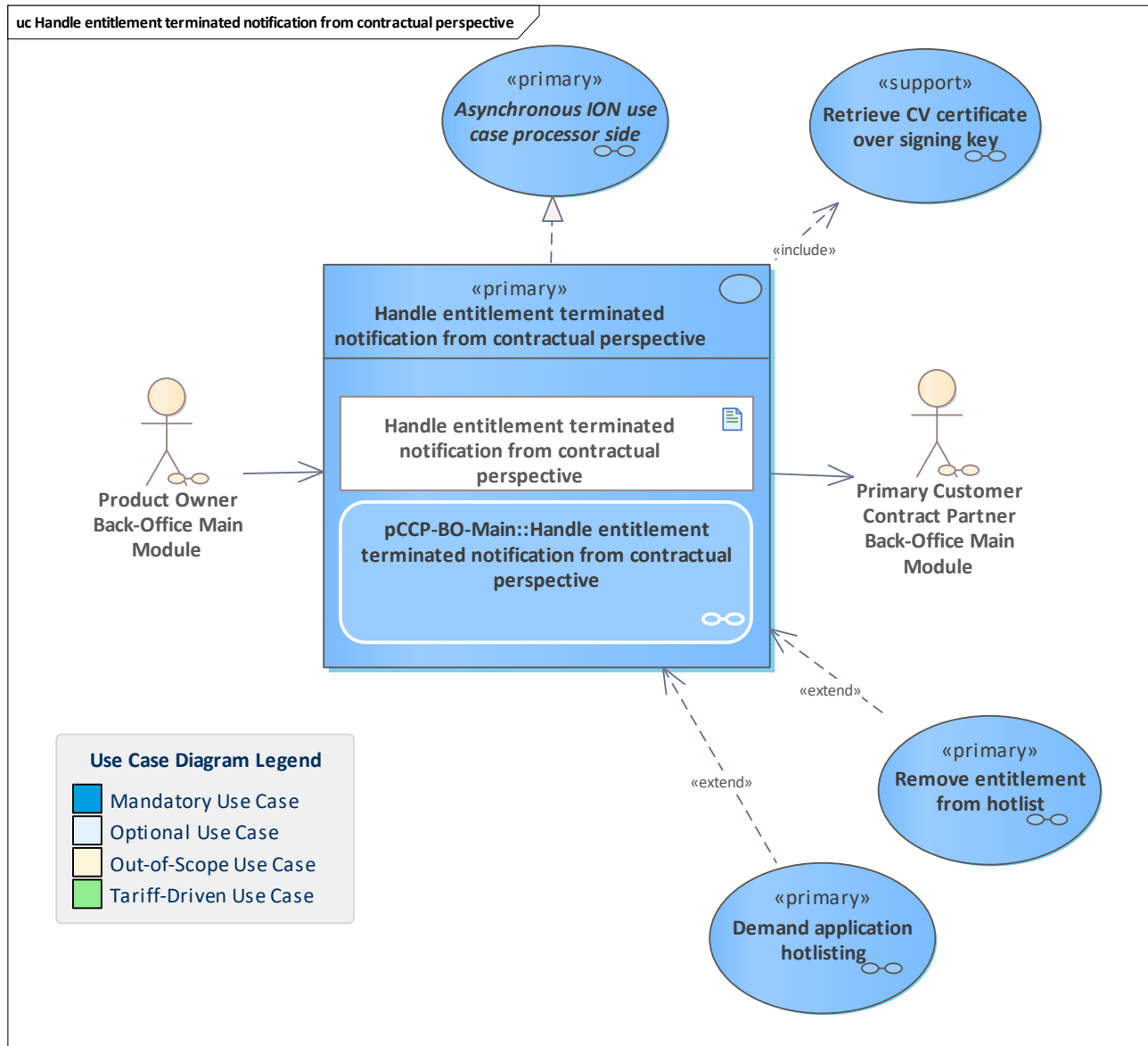
<b>Use Case</b>	<a href="#">Handle entitlement terminated notification from operational perspective</a>
<b>Description</b>	<p>Handle an entitlement terminated notification from the operational perspective.</p> <p>The entitlement terminated notification is sent by the CCP terminal to the CCP back-office system. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation.</p> <p><u>If the pCCP has terminated its own entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO</li> <li>the pCCP does its contractual checks and monitoring</li> </ul> <p><u>If a sCCP has terminated the entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO (and the PO will forward it later to the pCCP)</li> </ul> <p>In the case of action abortion, terminal and SAM action data is sent</p>



	by the terminal to the back-office system. The CCP back-office system registers the abortion for internal monitoring and data consistency.
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle entitlement terminated notification from contractual perspective</a> / <a href="#">Handle entitlement terminated notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement terminated notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">t Notify Entitlement Terminated: tNotifyEntitlementTerminated</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement terminated notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify entitlement terminating aborted : tNotifyEntitlementTerminatingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement terminating aborted notification from operational perspective</a>



## 14.2.4 Handle entitlement terminated notification from contractual perspective

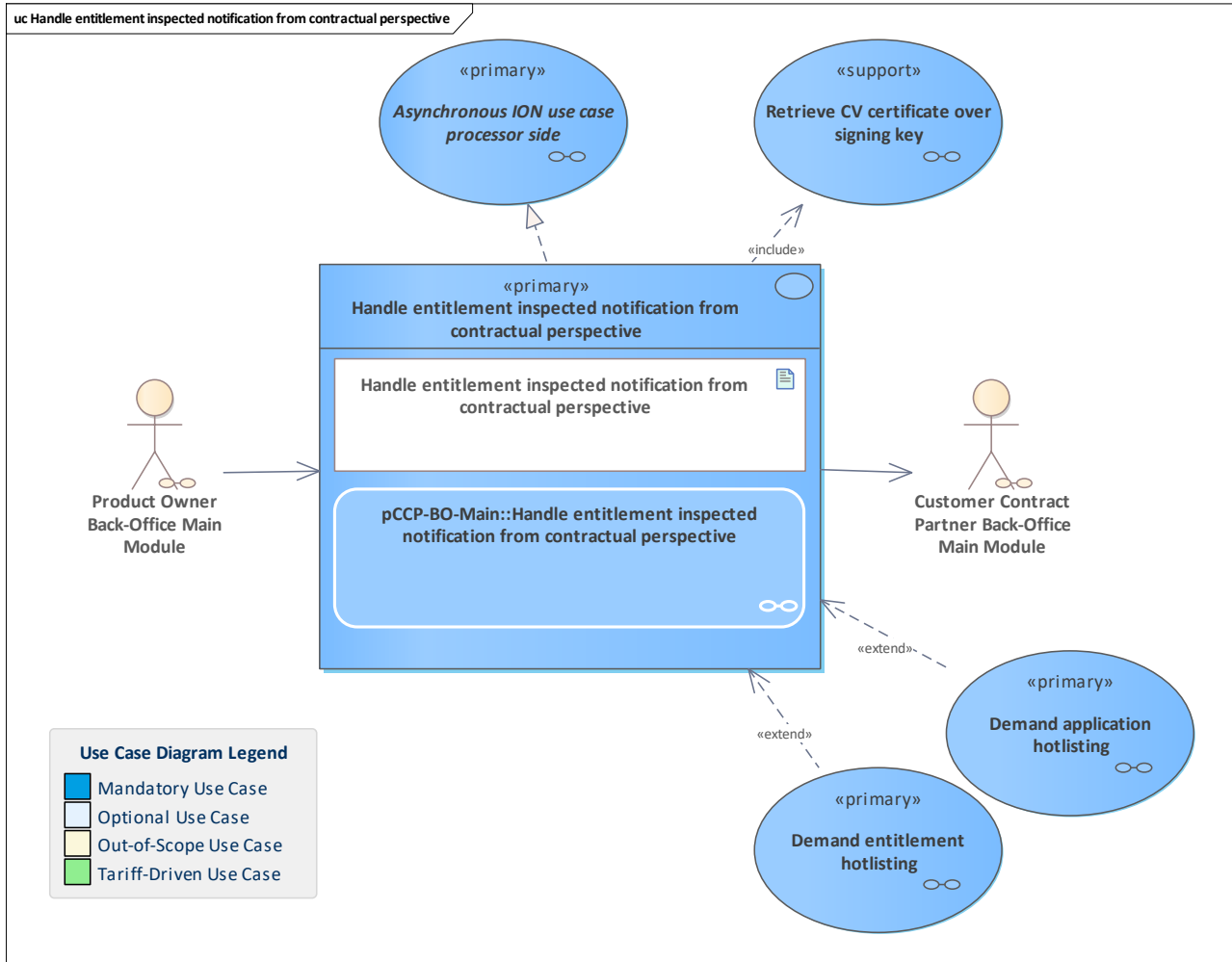


<b>Use Case</b>	<a href="#">Handle entitlement terminated notification from contractual perspective</a>
<b>Description</b>	<p>Handle an entitlement terminated notification from the contractual perspective.</p> <p>The entitlement terminated notification is received by the pCCP. The pCCP does its checks and monitoring from the contractual perspective regarding the correct execution of termination. In this context, the signature of the termination attestation is verified.</p> <p><b>Note:</b> this closes a potentially related user account concerning the entitlement.</p> <p>If the termination was correct, the pCCP checks if the entitlement has to be removed from the hotlist.</p> <p><b>Note:</b> if the pCCP itself performed the termination, this use case</p>



	takes place inside the use case <a href="#">Handle entitlement terminated notification from operational perspective</a> and is not called by the PO. In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a> .
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Remove entitlement from hotlist</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward entitlement terminated notification : forwardEntitlementTerminatedNotification</a>
<b>Outputs</b>	<a href="#">Forward entitlement terminated notification response : forwardEntitlementTerminatedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward entitlement terminated notification exception : forwardEntitlementTerminatedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle entitlement terminated notification from contractual perspective</a>

## 14.2.5 Handle entitlement inspected notification from contractual perspective



<b>Use Case</b>	<a href="#">Handle entitlement inspected notification from contractual perspective</a>
<b>Description</b>	This use case describes the processing of the notification of an inspected entitlement in the back-office system of the pCCP. The pCCP receives the notification from the PO system, registers it and does its contractual monitoring checks.
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand entitlement hotlisting</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	



<b>Inputs</b>	<u>Forward entitlement inspected notification :</u> <u>forwardEntitlementInspectedNotification</u>
<b>Outputs</b>	<u>Forward entitlement inspected notification response :</u> <u>forwardEntitlementInspectedNotificationResponse</u>
<b>Error Cases</b>	<u>E CCP RECEIVER IS NOT ENTITY OWNER</u> <u>E CO BINARY STRUCTURE CANNOT BE PROCESSED</u> <u>E CO WRONG ATTESTATION IN NOTIFICATION</u> <u>E CO WRONG SECURITY LEVEL</u> <u>E CO DUPLICATE UM ATTESTATION</u> <u>Forward entitlement inspected notification exception :</u> <u>forwardEntitlementInspectedNotificationException</u>
<b>Activity Diagram</b>	<u>pCCP-BO-Main::Handle entitlement inspected notification from contractual perspective</u>

# 15 Account-Based Payment Bundle CCP-System

Functionality bundle that covers the use cases for a CCP back-office system forming the basis for using the account-based payment method.

## 15.1 Overview

Handle entitlement issued notification from operational perspective

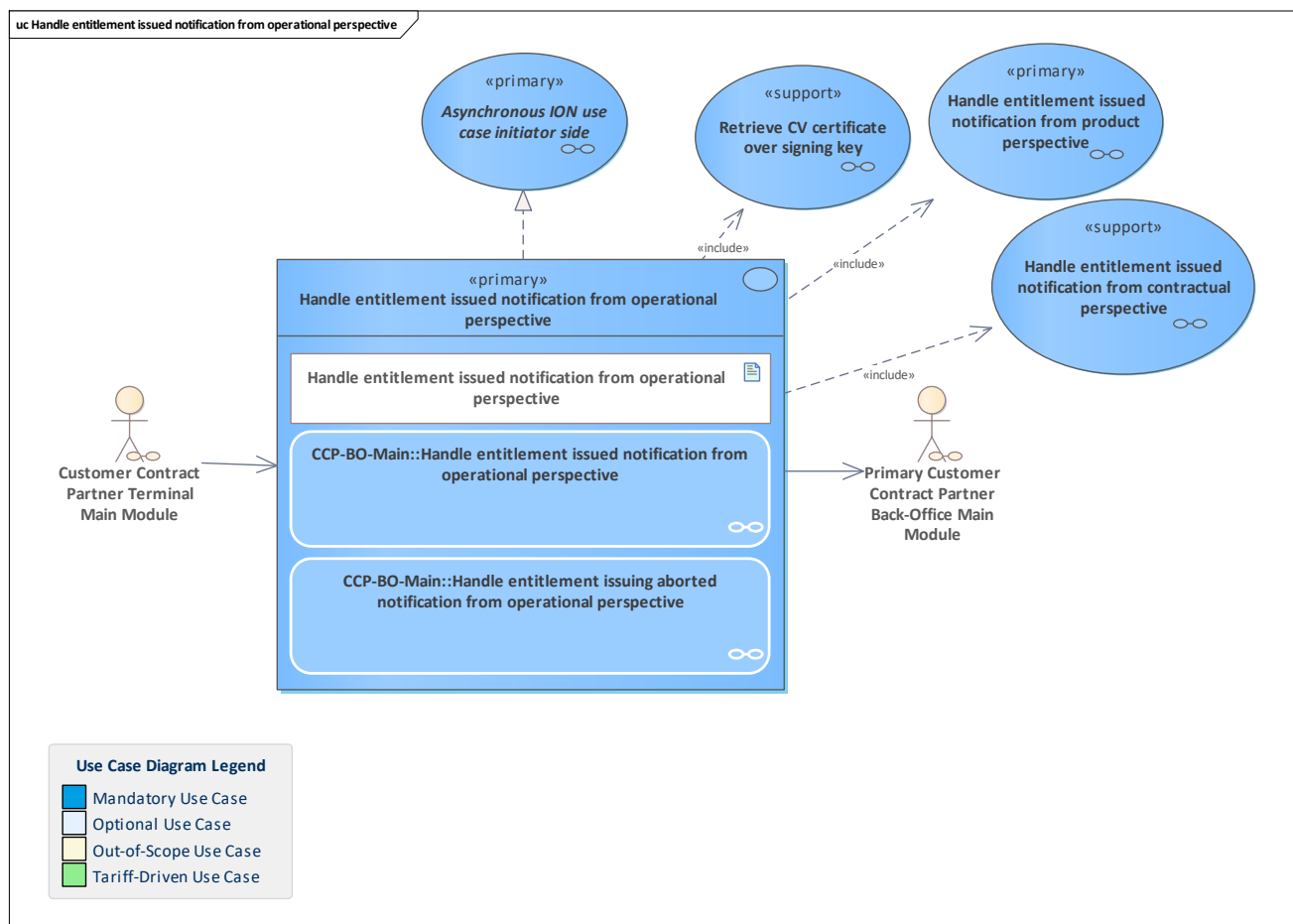
Handle entitlement issued notification from contractual perspective

Handle entitlement terminated notification from operational perspective

Handle entitlement terminated notification from contractual perspective

## 15.2 Use Cases

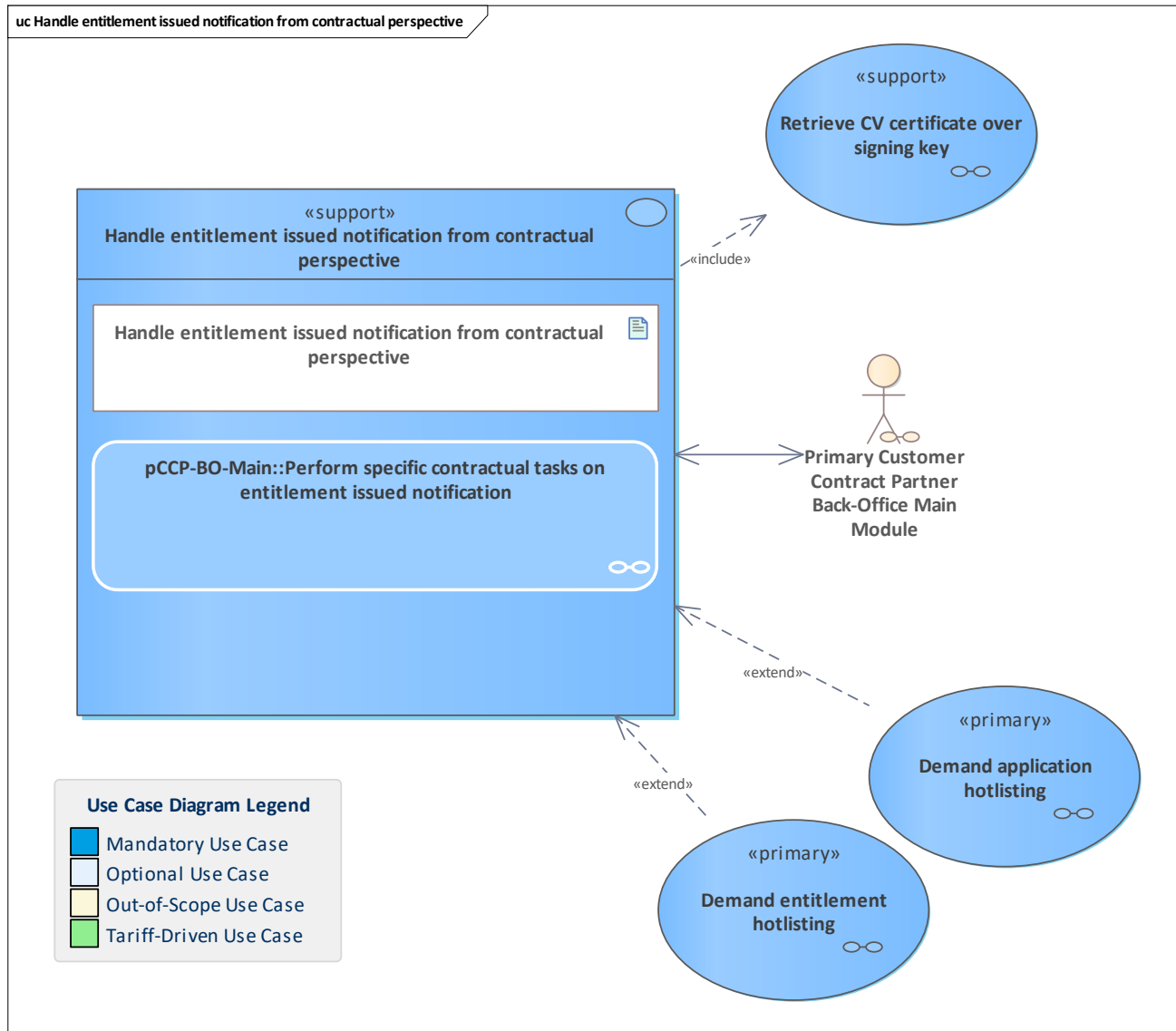
### 15.2.1 Handle entitlement issued notification from operational perspective





<b>Description</b>	<p>The pCCP of the entitlement receives the notification about an entitlement issuance and registers it.</p> <p>Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the issuance attestation.</p> <p>Since it is the pCCP, it also does the contractual checks and monitoring.</p> <p>Finally, the notification is forwarded to the responsible PO system. This can be done either with a single message or in a scheduled process with a list of messages.</p> <p>In case of an abortion, the notification is also sent to the PO system to announce the used SAM- and product issuance counters.</p> <p><b>Note:</b> this use case is without the context of ordered action execution.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement issued notification from product perspective</a> / <a href="#">Handle entitlement issued notification from contractual perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify entitlement issuing aborted : tNotifyEntitlementIssuingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement issuing aborted notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify entitlement issued : tNotifyEntitlementIssued</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement issued notification from operational perspective</a>

## 15.2.2 Handle entitlement issued notification from contractual perspective



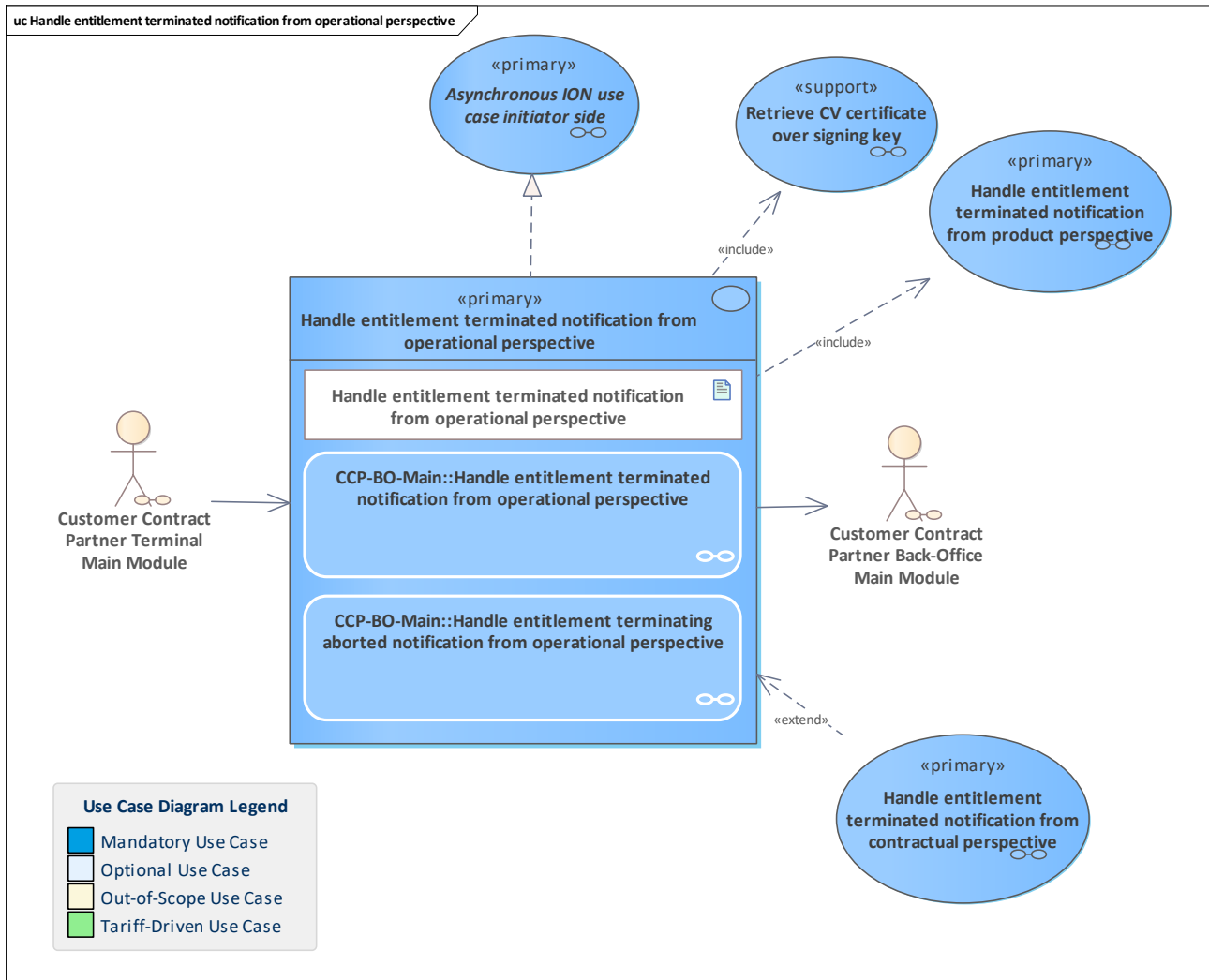
<b>Use Case</b>	<a href="#">Handle entitlement issued notification from contractual perspective</a>
<b>Description</b>	Handle a notification about the issuance of an owned entitlement from the contractual perspective by performing the related checks and monitoring. Note that the application instance ID of the user medium the entitlement was issued to can be uniquely identified via <i>umAppInstanceId</i> , which is part of the <a href="#">SignedEntitlementIssuedAttestation</a> that is contained in the <a href="#">EntitlementIssuedNotification</a> .
<b>Initiating Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	<a href="#">Demand entitlement hotlisting</a> / <a href="#">Demand application hotlisting</a>



<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Perform specific contractual tasks on entitlement issued notification</a>



## 15.2.3 Handle entitlement terminated notification from operational perspective

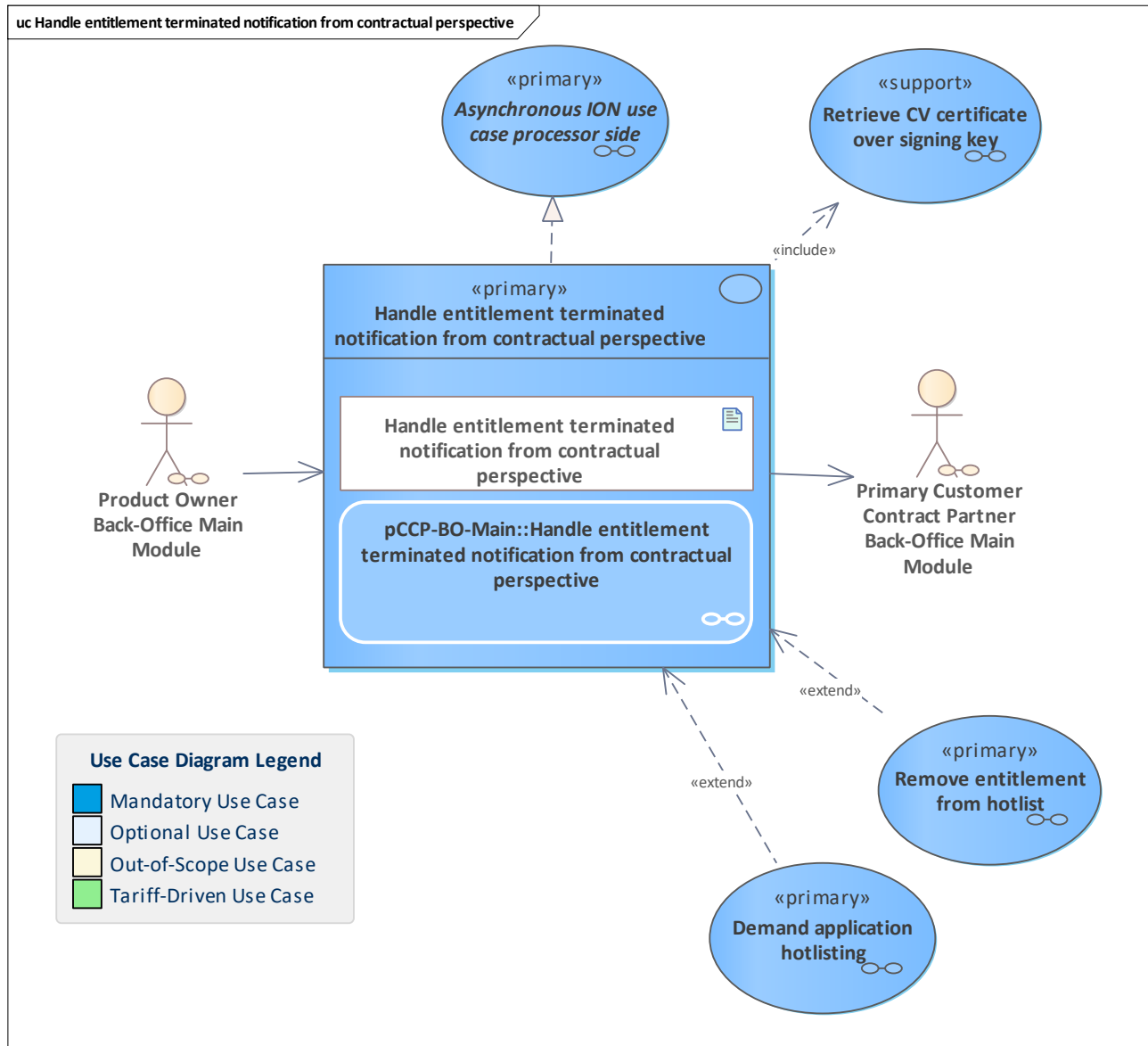


<b>Use Case</b>	<a href="#">Handle entitlement terminated notification from operational perspective</a>
<b>Description</b>	<p>Handle an entitlement terminated notification from the operational perspective.</p> <p>The entitlement terminated notification is sent by the CCP terminal to the CCP back-office system. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation.</p> <p><u>If the pCCP has terminated its own entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO</li> <li>the pCCP does its contractual checks and monitoring</li> </ul> <p><u>If a sCCP has terminated the entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO (and the PO will forward it later to the pCCP)</li> </ul> <p>In the case of action abortion, terminal and SAM action data is sent</p>



	by the terminal to the back-office system. The CCP back-office system registers the abortion for internal monitoring and data consistency.
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle entitlement terminated notification from contractual perspective</a> / <a href="#">Handle entitlement terminated notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement terminated notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">t Notify Entitlement Terminated: tNotifyEntitlementTerminated</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement terminated notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify entitlement terminating aborted : tNotifyEntitlementTerminatingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement terminating aborted notification from operational perspective</a>

## 15.2.4 Handle entitlement terminated notification from contractual perspective



<b>Use Case</b>	<u>Handle entitlement terminated notification from contractual perspective</u>
<b>Description</b>	<p>Handle an entitlement terminated notification from the contractual perspective.</p> <p>The entitlement terminated notification is received by the pCCP. The pCCP does its checks and monitoring from the contractual perspective regarding the correct execution of termination. In this context, the signature of the termination attestation is verified.</p> <p><b>Note:</b> this closes a potentially related user account concerning the entitlement.</p> <p>If the termination was correct, the pCCP checks if the entitlement has to be removed from the hotlist.</p> <p><b>Note:</b> if the pCCP itself performed the termination, this use case</p>



	takes place inside the use case <a href="#">Handle entitlement terminated notification from operational perspective</a> and is not called by the PO. In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a> .
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Remove entitlement from hotlist</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward entitlement terminated notification : forwardEntitlementTerminatedNotification</a>
<b>Outputs</b>	<a href="#">Forward entitlement terminated notification response : forwardEntitlementTerminatedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward entitlement terminated notification exception : forwardEntitlementTerminatedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle entitlement terminated notification from contractual perspective</a>



# 16 Stored-Value Payment Bundle CCP-System

Functionality bundle that covers the use cases for a CCP back-office system forming the basis for using the stored-value payment method.

## 16.1 Overview

Handle entitlement issued notification from operational perspective

Handle entitlement issued notification from contractual perspective

Handle entitlement terminated notification from operational perspective

Handle entitlement terminated notification from contractual perspective

Handle stored-value payment method recharged notification from operational perspective

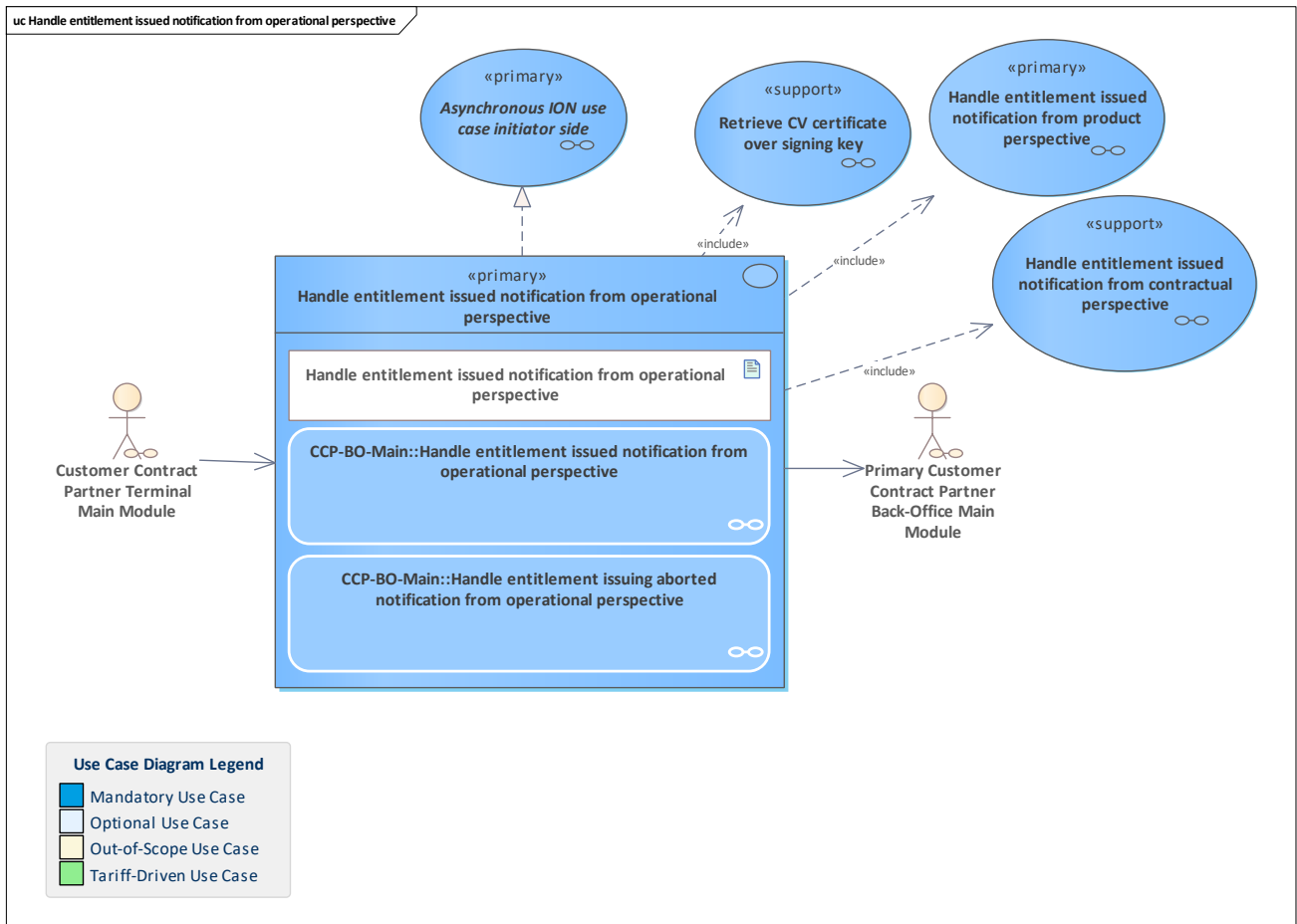
Handle stored-value payment method recharged notification from contractual perspective

Handle stored-value payment method reimbursed notification from operational perspective

Handle stored-value payment method reimbursed notification from contractual perspective

## 16.2 Use Cases

### 16.2.1 Handle entitlement issued notification from operational perspective

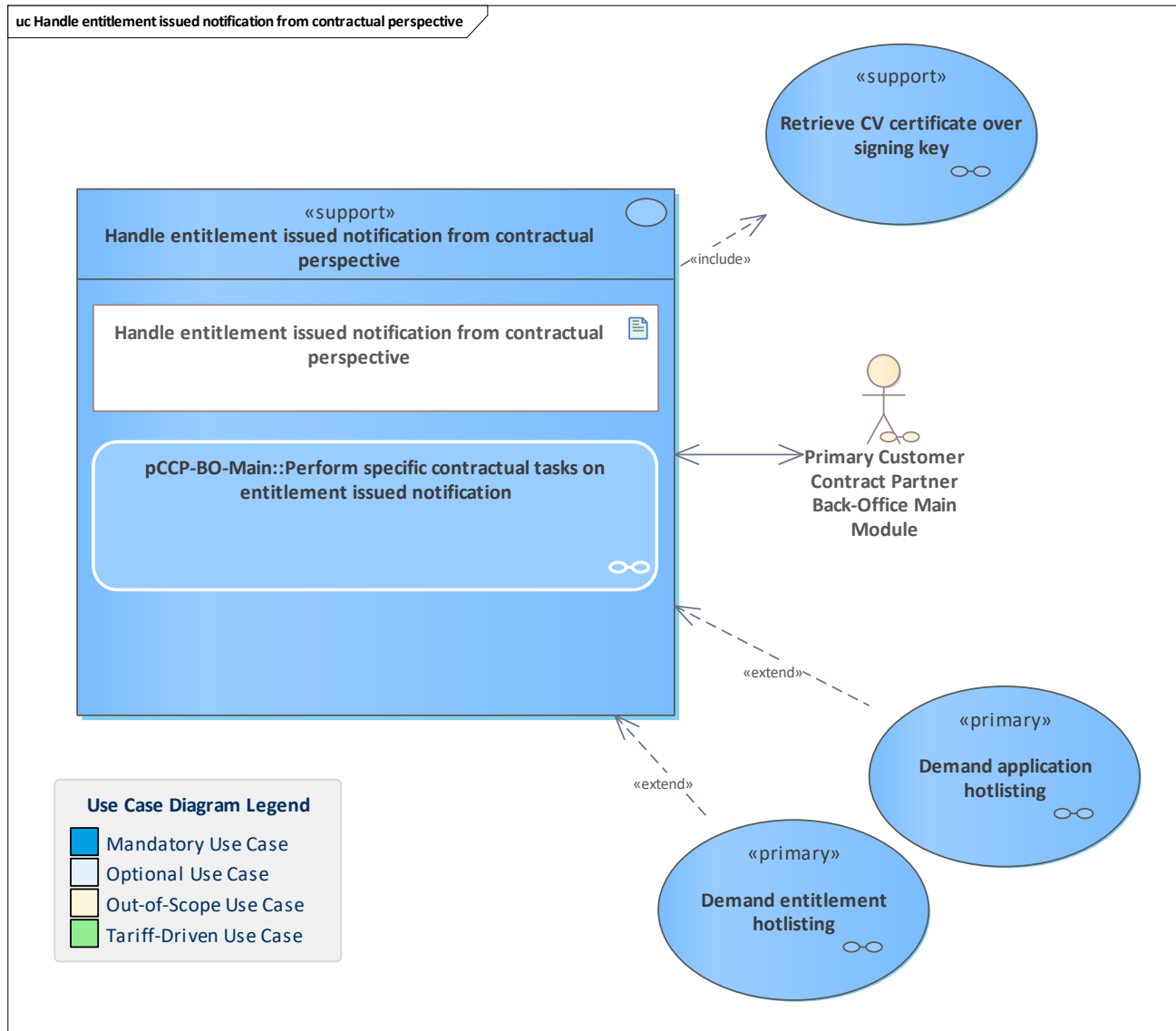


<b>Use Case</b>	<a href="#">Handle entitlement issued notification from operational perspective</a>
<b>Description</b>	<p>The pCCP of the entitlement receives the notification about an entitlement issuance and registers it.</p> <p>Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the issuance attestation.</p> <p>Since it is the pCCP, it also does the contractual checks and monitoring.</p> <p>Finally, the notification is forwarded to the responsible PO system. This can be done either with a single message or in a scheduled process with a list of messages.</p> <p>In case of an abortion, the notification is also sent to the PO system to announce the used SAM- and product issuance counters.</p> <p><b>Note:</b> this use case is without the context of ordered action execution.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement issued notification from product perspective</a> / <a href="#">Handle entitlement issued notification from contractual perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate</a>



	<a href="#">over signing key / Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify entitlement issuing aborted : tNotifyEntitlementIssuingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement issuing aborted notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify entitlement issued : tNotifyEntitlementIssued</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement issued notification from operational perspective</a>

## 16.2.2 Handle entitlement issued notification from contractual perspective



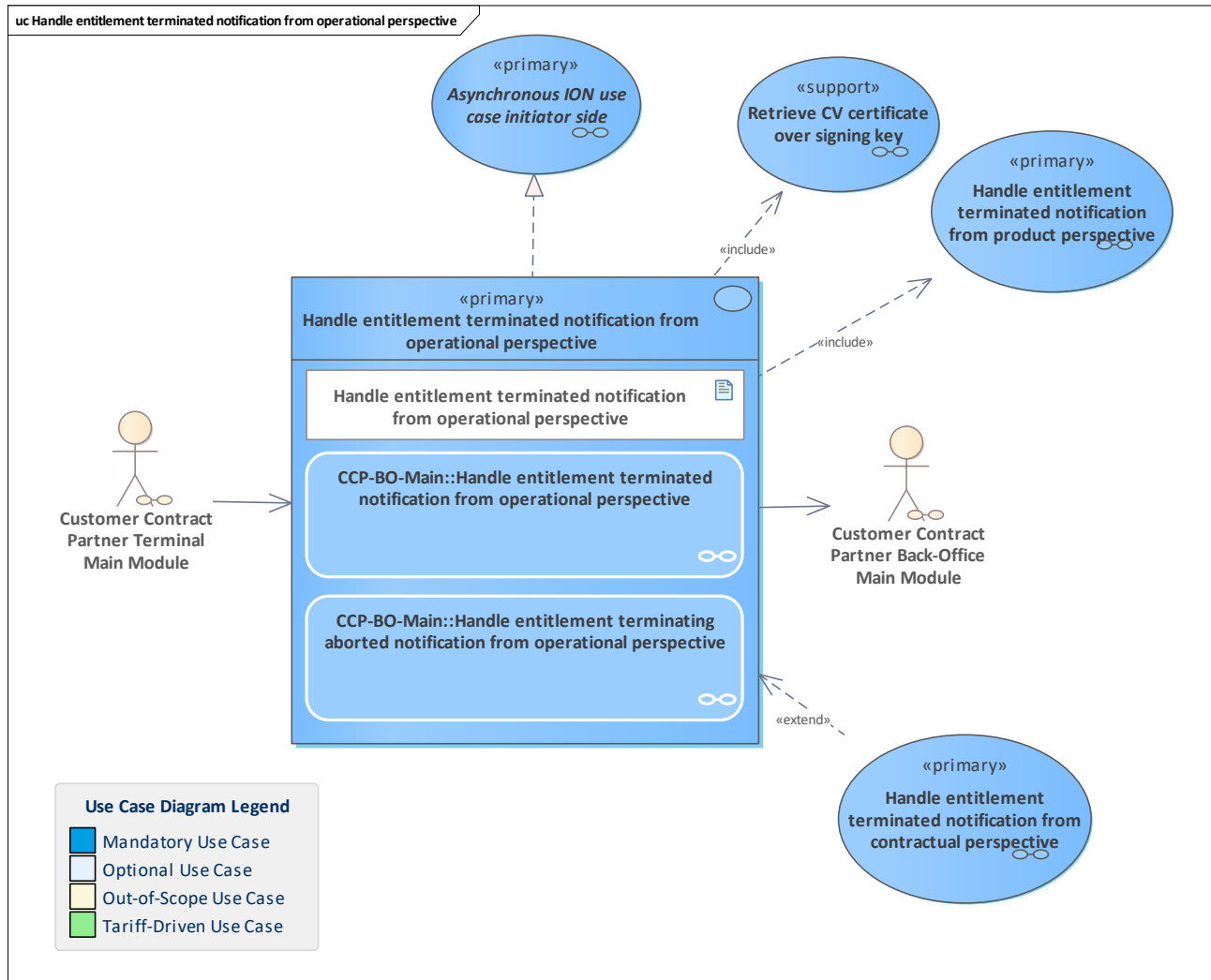
<b>Use Case</b>	<a href="#">Handle entitlement issued notification from contractual perspective</a>
<b>Description</b>	Handle a notification about the issuance of an owned entitlement from the contractual perspective by performing the related checks and monitoring. Note that the application instance ID of the user medium the entitlement was issued to can be uniquely identified via <i>umAppInstanceId</i> , which is part of the <a href="#">SignedEntitlementIssuedAttestation</a> that is contained in the <a href="#">EntitlementIssuedNotification</a> .
<b>Initiating Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	<a href="#">Demand entitlement hotlisting</a> / <a href="#">Demand application hotlisting</a>





<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Perform specific contractual tasks on entitlement issued notification</a>

## 16.2.3 Handle entitlement terminated notification from operational perspective

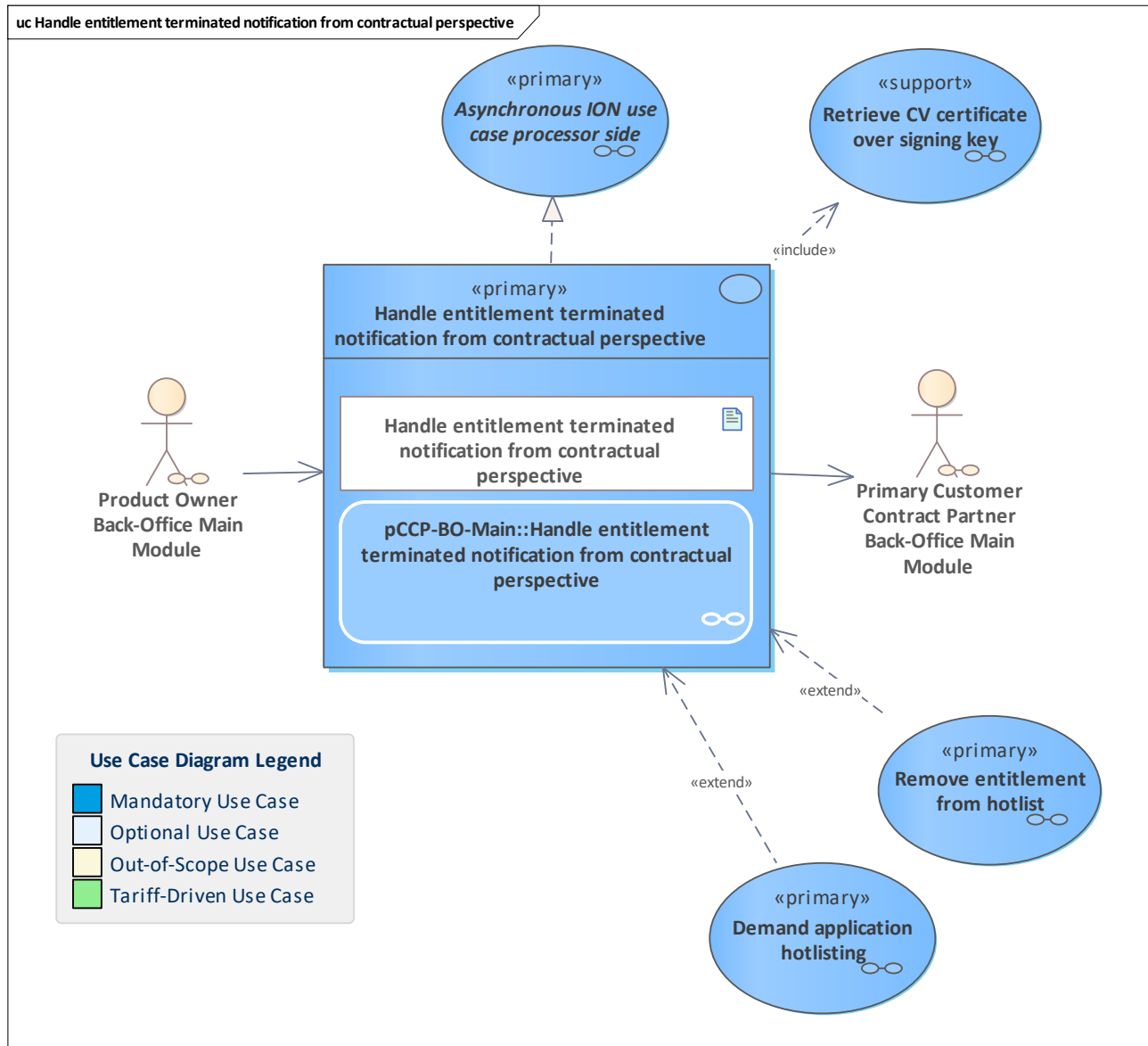


<b>Use Case</b>	<a href="#">Handle entitlement terminated notification from operational perspective</a>
<b>Description</b>	<p>Handle an entitlement terminated notification from the operational perspective.</p> <p>The entitlement terminated notification is sent by the CCP terminal to the CCP back-office system. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation.</p> <p><u>If the pCCP has terminated its own entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO</li> <li>the pCCP does its contractual checks and monitoring</li> </ul> <p><u>If a sCCP has terminated the entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO (and the PO will forward it later to the pCCP)</li> </ul> <p>In the case of action abortion, terminal and SAM action data is sent</p>



	by the terminal to the back-office system. The CCP back-office system registers the abortion for internal monitoring and data consistency.
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle entitlement terminated notification from contractual perspective</a> / <a href="#">Handle entitlement terminated notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement terminated notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">t Notify Entitlement Terminated: tNotifyEntitlementTerminated</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement terminated notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify entitlement terminating aborted : tNotifyEntitlementTerminatingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle entitlement terminating aborted notification from operational perspective</a>

## 16.2.4 Handle entitlement terminated notification from contractual perspective

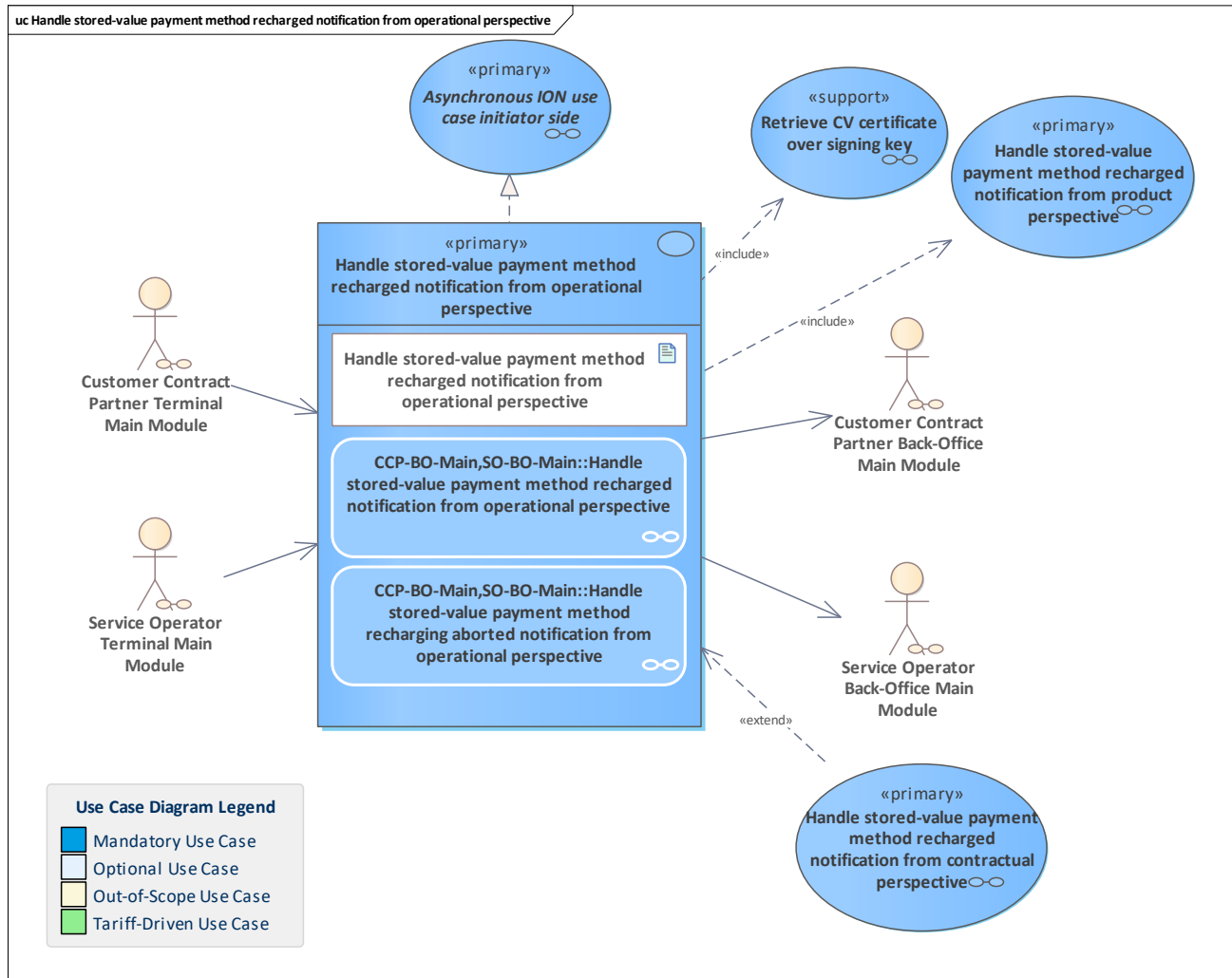


<b>Use Case</b>	<u>Handle entitlement terminated notification from contractual perspective</u>
<b>Description</b>	<p>Handle an entitlement terminated notification from the contractual perspective.</p> <p>The entitlement terminated notification is received by the pCCP. The pCCP does its checks and monitoring from the contractual perspective regarding the correct execution of termination. In this context, the signature of the termination attestation is verified.</p> <p><b>Note:</b> this closes a potentially related user account concerning the entitlement.</p> <p>If the termination was correct, the pCCP checks if the entitlement has to be removed from the hotlist.</p> <p><b>Note:</b> if the pCCP itself performed the termination, this use case</p>



	takes place inside the use case <a href="#">Handle entitlement terminated notification from operational perspective</a> and is not called by the PO. In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a> .
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Remove entitlement from hotlist</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward entitlement terminated notification : forwardEntitlementTerminatedNotification</a>
<b>Outputs</b>	<a href="#">Forward entitlement terminated notification response : forwardEntitlementTerminatedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward entitlement terminated notification exception : forwardEntitlementTerminatedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle entitlement terminated notification from contractual perspective</a>

## 16.2.5 Handle stored-value payment method recharged notification from operational perspective

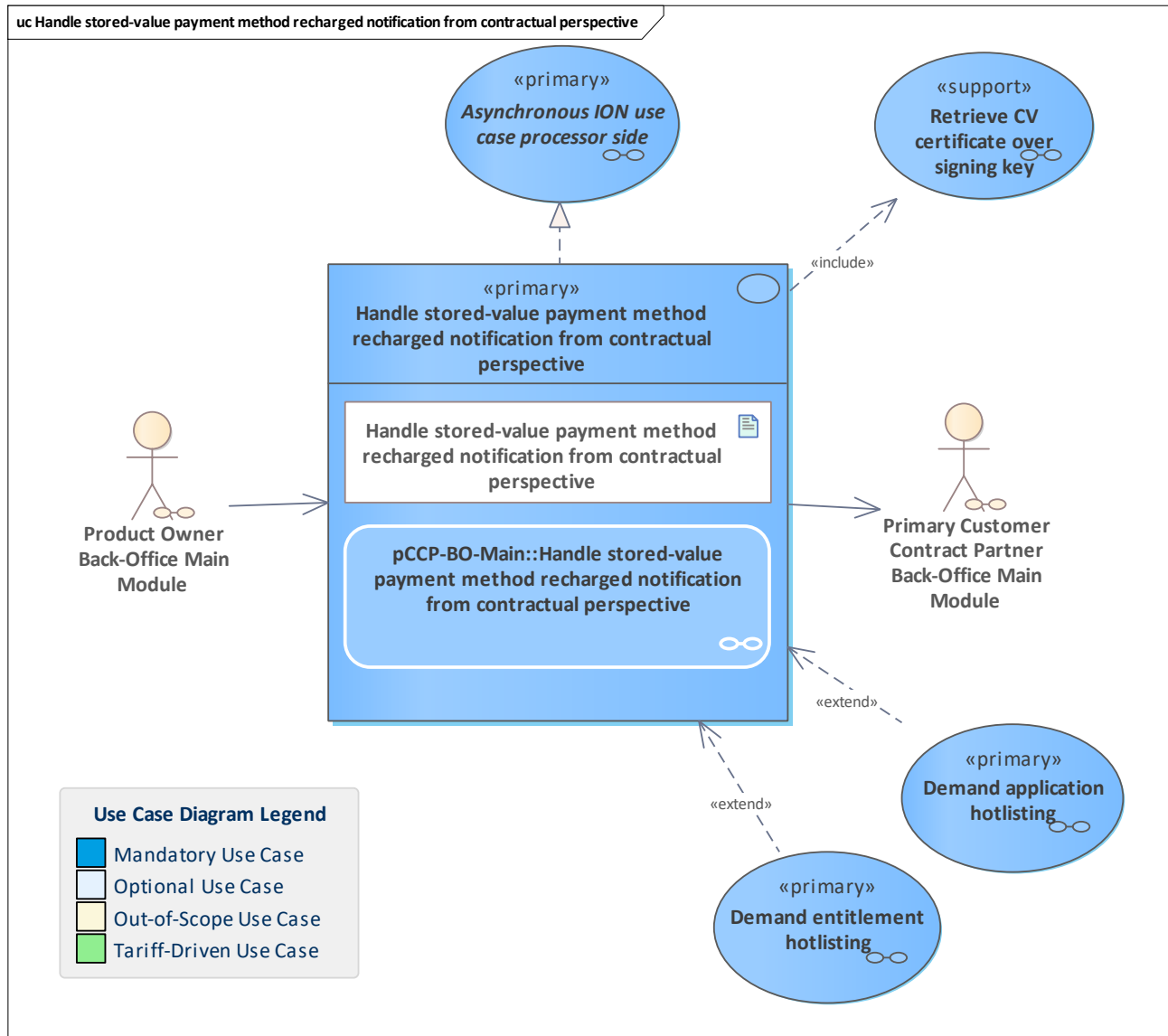


<b>Use Case</b>	<a href="#">Handle stored-value payment method recharged notification from operational perspective</a>
<b>Description</b>	<p>Handle a stored-value payment method recharged notification from the operational perspective.</p> <p>The CCP back-office system receives the notification about the recharge action of a stored-value payment method and registers it. Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the recharge action attestation.</p> <p>Then, the notification is forwarded to the responsible PO system. If the current CCP is the pCCP, it can also do the contractual checks and monitoring directly.</p> <p>In the case of a transaction abortion, the notification is registered for consistent monitoring. Since no counters are affected which are important to the PO, the notification is not forwarded.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a> <a href="#">Service Operator Terminal Main Module</a>



<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle stored-value payment method recharged notification from contractual perspective</a> / <a href="#">Handle stored-value payment method recharged notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle stored-value payment method recharged notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify stored-value payment method recharged : tNotifyStoredValuePaymentMethodRecharged</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main,SO-BO-Main::Handle stored-value payment method recharged notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify stored-value payment method recharging aborted : tNotifyStoredValuePaymentMethodRechargingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main,SO-BO-Main::Handle stored-value payment method recharging aborted notification from operational perspective</a>

## 16.2.6 Handle stored-value payment method recharged notification from contractual perspective



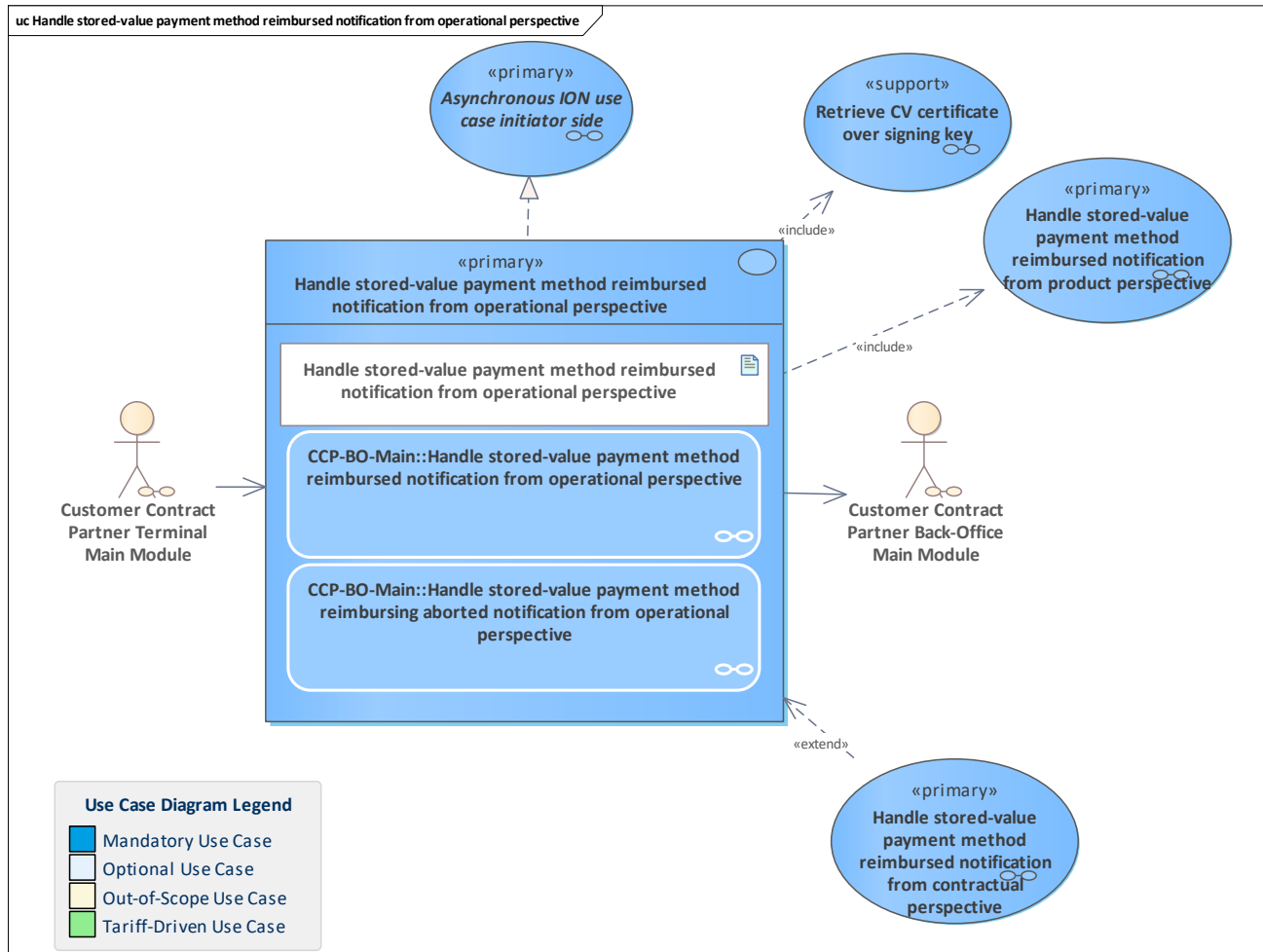
<b>Use Case</b>	<a href="#">Handle stored-value payment method recharged notification from contractual perspective</a>
<b>Description</b>	<p>Handle a stored-value payment method recharged notification from the contractual perspective.</p> <p>The pCCP does its contractual checks and monitoring.</p> <p><b>Note:</b> if the pCCP itself performed the recharge action, this use case takes place inside the use case <a href="#">Handle stored-value payment method recharged notification from operational perspective</a> and is not called by the PO.</p> <p>In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a>.</p>
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>





<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Demand entitlement hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward stored-value payment method recharged notification : forwardStoredValuePaymentMethodRechargedNotification</a>
<b>Outputs</b>	<a href="#">Forward stored-value payment method recharged notification response : forwardStoredValuePaymentMethodRechargedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward stored-value payment method recharged notification exception : forwardStoredValuePaymentMethodRechargedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle stored-value payment method recharged notification from contractual perspective</a>

## 16.2.7 Handle stored-value payment method reimbursed notification from operational perspective

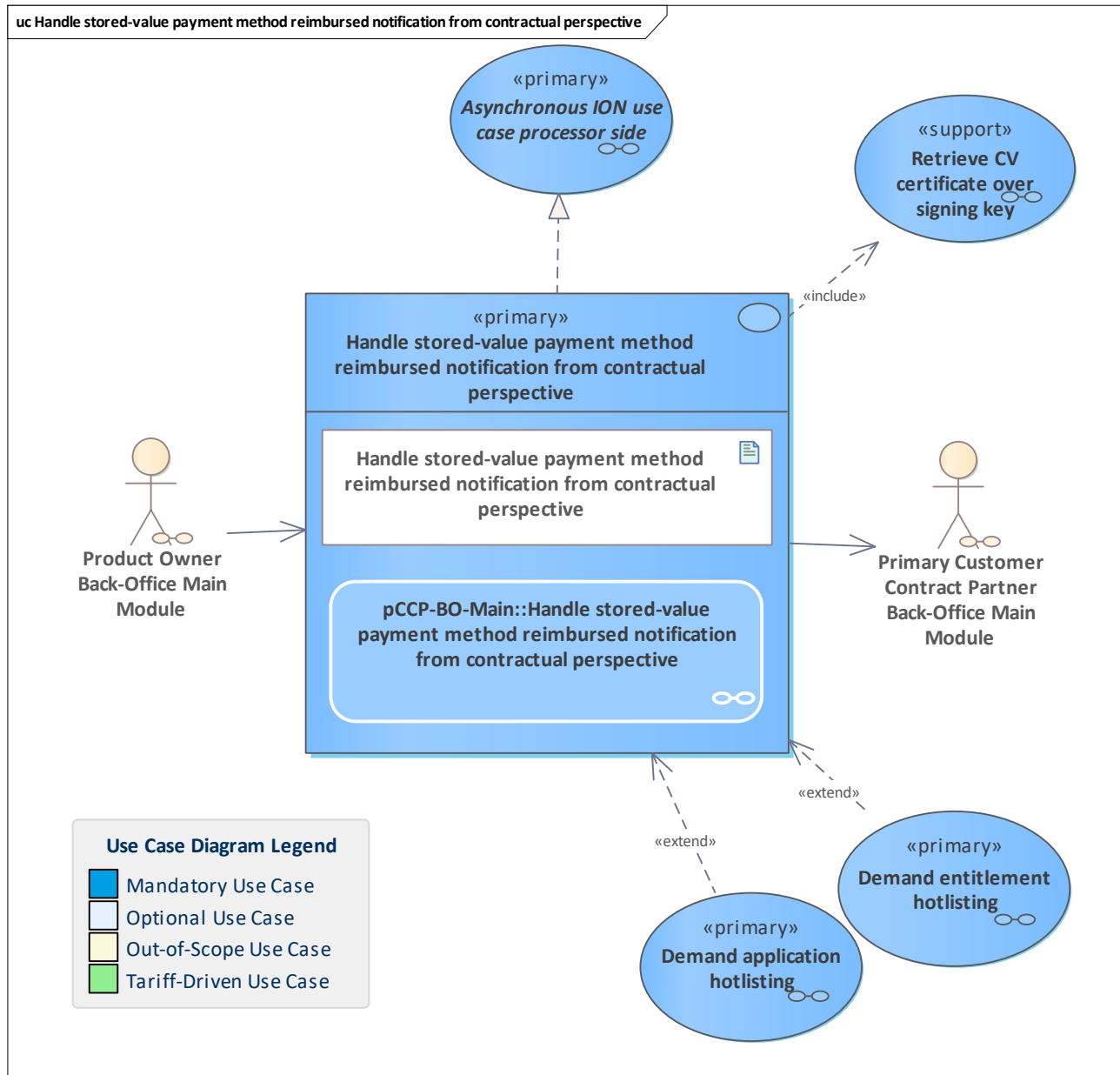


<b>Use Case</b>	<a href="#">Handle stored-value payment method reimbursed notification from operational perspective</a>
<b>Description</b>	<p>Handle a stored-value payment method reimbursed notification from the operational perspective.</p> <p>The CCP back-office system receives the notification about the reimburse action of an stored-value payment method and registers it.</p> <p>Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the reimburse action attestation.</p> <p>Then, the notification is forwarded to the responsible PO system.</p> <p>If the current CCP is the pCCP, it can also do the contractual checks and monitoring directly.</p> <p>In the case of a transaction abortion, the notification is registered for consistent monitoring. Since no counters are affected which are important to the PO, the notification is not forwarded.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	



<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle stored-value payment method reimbursed notification from contractual perspective</a> / <a href="#">Handle stored-value payment method reimbursed notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle stored-value payment method reimbursed notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify stored-value payment method reimbursed : tNotifyStoredValuePaymentMethodReimbursed</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle stored-value payment method reimbursed notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify stored-value payment method reimbursing aborted : tNotifyStoredValuePaymentMethodReimbursingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle stored-value payment method reimbursing aborted notification from operational perspective</a>

## 16.2.8 Handle stored-value payment method reimbursed notification from contractual perspective



<b>Use Case</b>	<a href="#">Handle stored-value payment method reimbursed notification from contractual perspective</a>
<b>Description</b>	<p>Handle a stored-value payment method reimbursed notification from the contractual perspective.</p> <p>The pCCP does its contractual checks and monitoring.</p> <p><b>Note:</b> if the pCCP itself performed the reimburse action, this use case takes place inside the use case <a href="#">Handle stored- value payment method reimbursed notification from operational perspective</a> and is not called by the PO.</p> <p>In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a>.</p>



<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand entitlement hotlisting</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward stored-value payment method reimbursed notification : forwardStoredValuePaymentMethodReimbursedNotification</a>
<b>Outputs</b>	<a href="#">Forward stored-value payment method reimbursed notification response : forwardStoredValuePaymentMethodReimbursedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward stored-value payment method reimbursed notification exception : forwardStoredValuePaymentMethodReimbursedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle stored-value payment method reimbursed notification from contractual perspective</a>



## 17 Sale Electronic Ticket via Account-Based Payment Bundle CCP-System

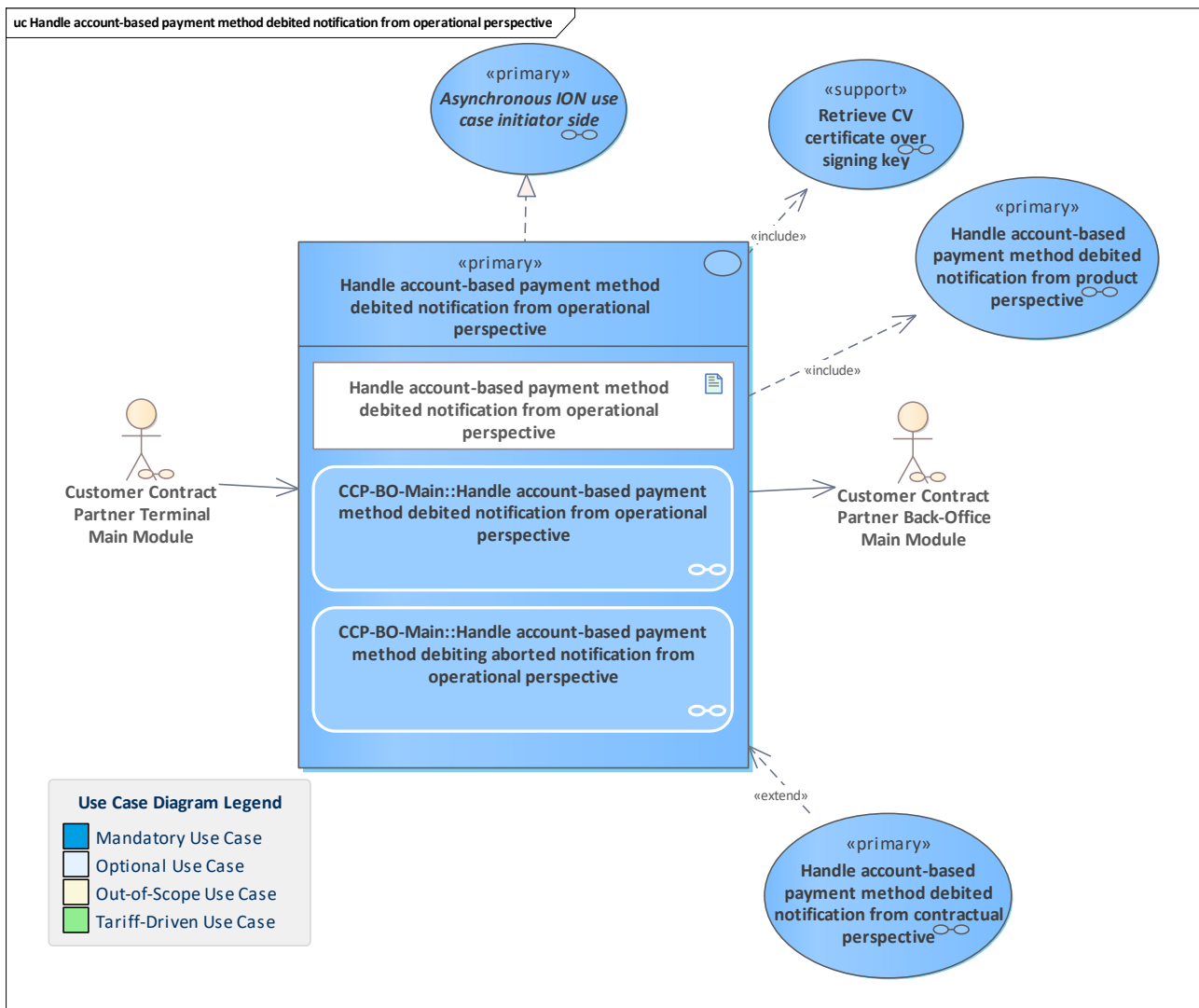
Functionality bundle that provides CCP back-office system use cases for selling or purchasing electronic tickets via an account-based payment method on a user medium.

### 17.1 Overview

Handle account-based payment method debited notification from operational perspective  
Handle account-based payment method debited notification from contractual perspective  
Handle account-based payment method credited notification from operational perspective  
Handle account-based payment method credited notification from contractual perspective

### 17.2 Use Cases

#### 17.2.1 Handle account-based payment method debited notification from operational perspective



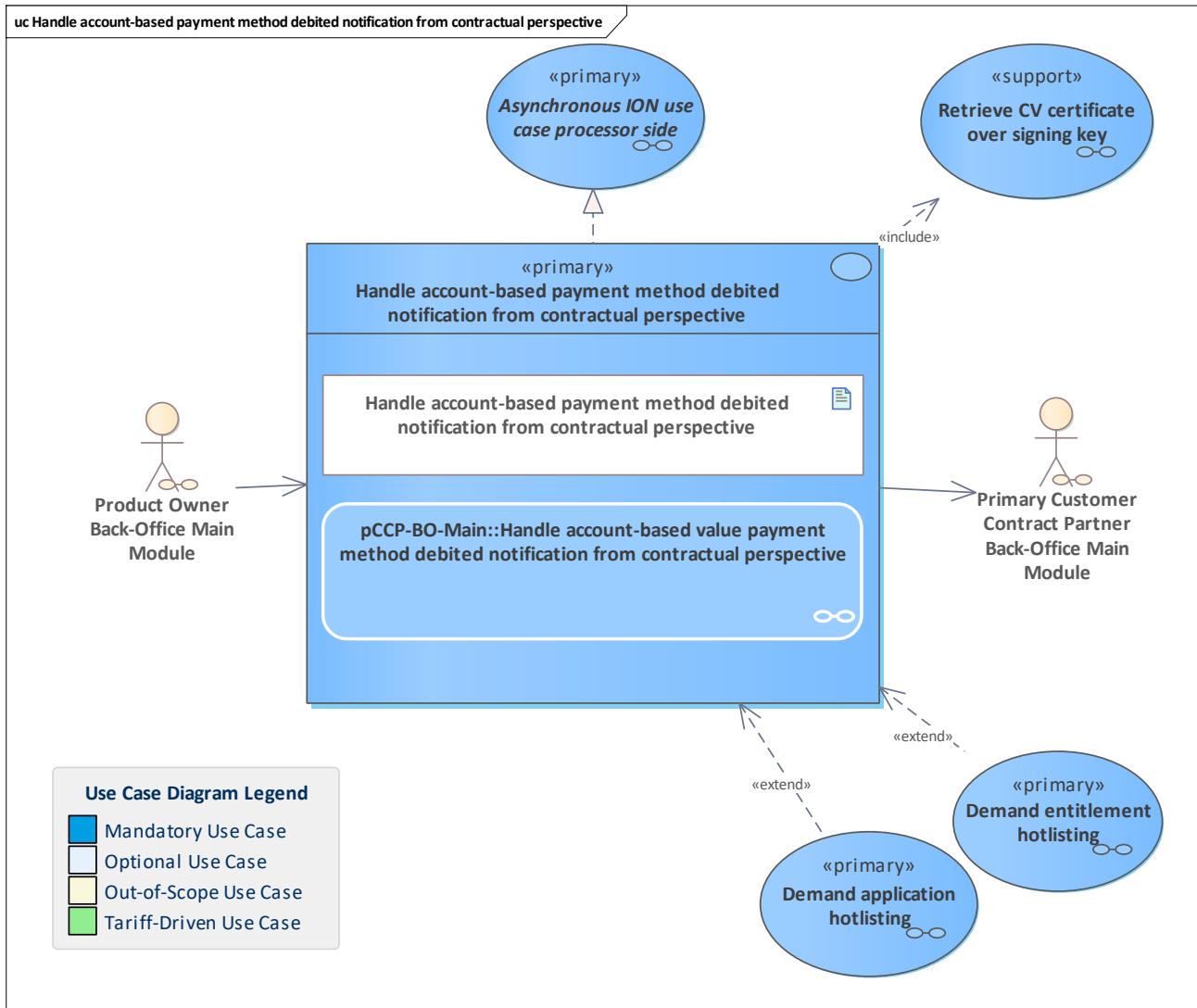
<b>Use Case</b>	<a href="#">Handle account-based payment method debited notification from operational perspective</a>
<b>Description</b>	<p>Handle a notification about an account-based payment method debiting from the operational perspective.</p> <p>The CCP back-office system receives the notification about the debit action of an account-based payment method and registers it. Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the debit action attestation.</p> <p>Then, the notification is forwarded to the responsible PO system. If the current CCP is the pCCP, it can also do the contractual checks and monitoring directly.</p> <p>In the case of an abortion, the notification is registered for consistent monitoring. Since no counters are affected which are important for the PO, the notification is not forwarded.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	<a href="#">Handle account-based payment method debited notification from</a>



<b>(Extended By)</b>	<a href="#">contractual perspective / Handle account-based payment method debited notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle account-based payment method debited notification from product perspective / Retrieve CV certificate over signing key / Retrieve CV certificate over signing key / Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side / Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify account-based payment method debited : tNotifyAccountBasedPaymentMethodDebited</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle account-based payment method debited notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify account-based payment method debiting aborted : tNotifyAccountBasedPaymentMethodDebitingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle account-based payment method debiting aborted notification from operational perspective</a>



## 17.2.2 Handle account-based payment method debited notification from contractual perspective

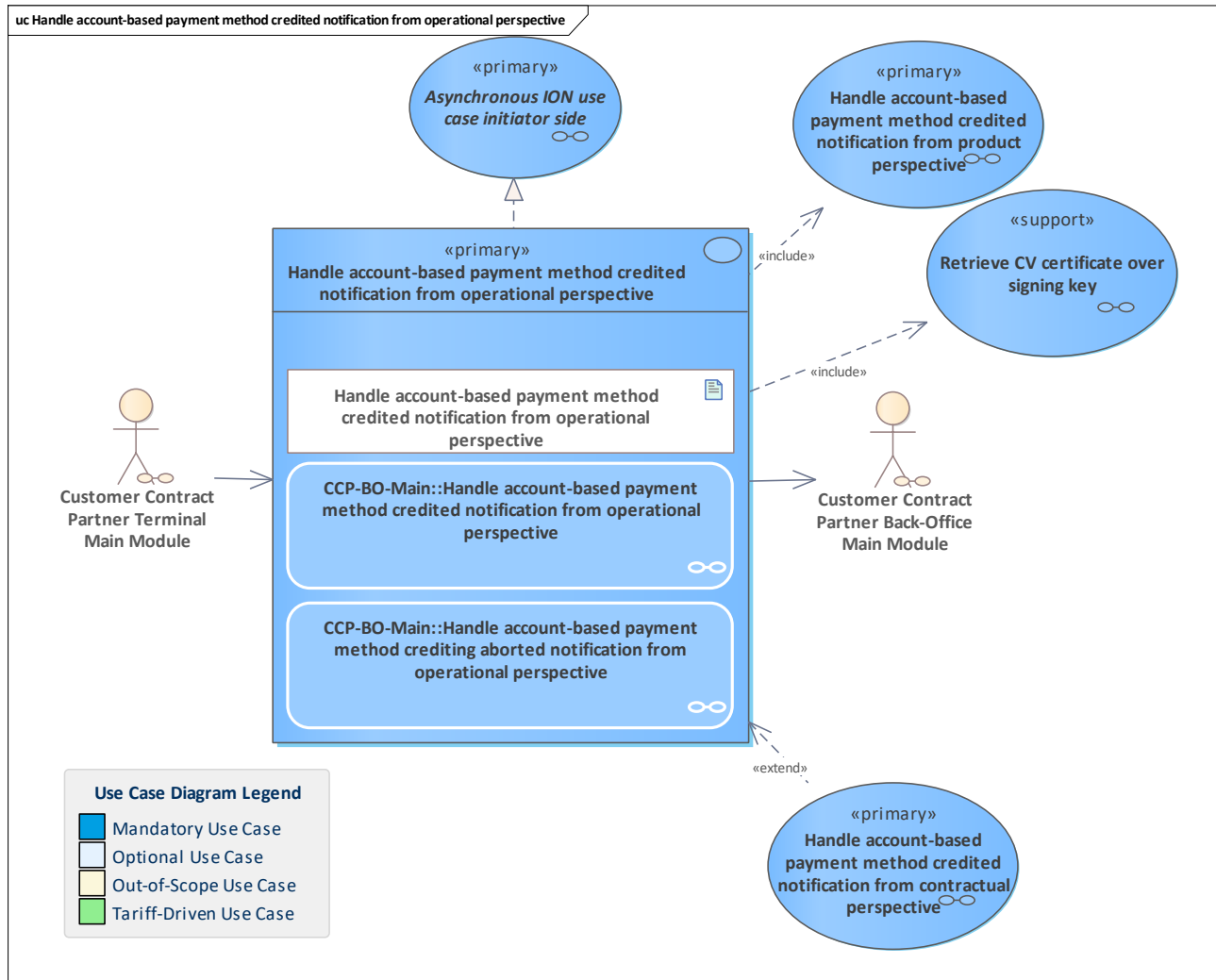


<b>Use Case</b>	<a href="#">Handle account-based payment method debited notification from contractual perspective</a>
<b>Description</b>	<p>Handle a notification about an account-based payment method debiting from the contractual perspective.</p> <p>The pCCP does its contractual checks and monitoring.</p> <p><b>Note:</b> if the pCCP itself performed the debit action, this use case takes place inside the use case <a href="#">Handle account-based payment method debited notification from operational perspective</a> and is not called by the PO.</p> <p>In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a>.</p>
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	



<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand entitlement hotlisting</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward account-based payment method debited notification : forwardAccountBasedPaymentMethodDebitedNotification</a>
<b>Outputs</b>	<a href="#">Forward account-based payment method debited notification response : forwardAccountBasedPaymentMethodDebitedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward account-based payment method debited notification exception : forwardAccountBasedPaymentMethodDebitedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle account-based value payment method debited notification from contractual perspective</a>

## 17.2.3 Handle account-based payment method credited notification from operational perspective

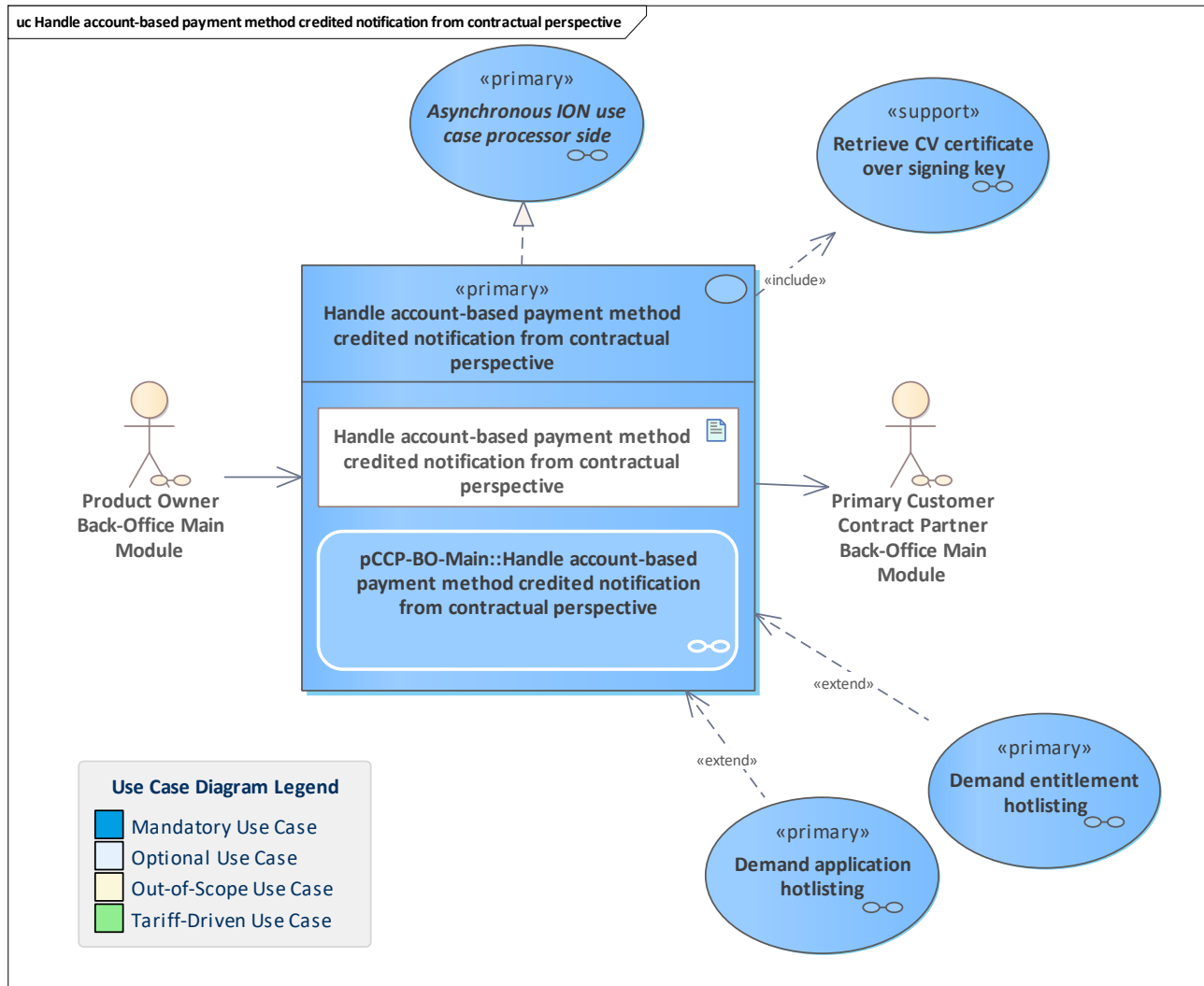


<b>Use Case</b>	<a href="#">Handle account-based payment method credited notification from operational perspective</a>
<b>Description</b>	<p>Handle an account-based payment method credited notification from the operational perspective.</p> <p>The CCP back-office system receives the notification about the credit transaction of an account-based payment method and registers it.</p> <p>Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the credit action attestation.</p> <p>Then, the notification is forwarded to the responsible PO system.</p> <p>If the current CCP is the pCCP, it can also do the contractual checks and monitoring directly.</p> <p>In the case of an abortion, the notification is registered for consistent monitoring. Since no counters are affected which are important for the PO, the notification is not forwarded.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>



<b>Reacting Actor</b>	<u>Customer Contract Partner Back-Office Main Module</u>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<u>Handle account-based payment method credited notification from contractual perspective</u> / <u>Handle account-based payment method credited notification from contractual perspective</u>
<b>Linked Use Cases (Includes)</b>	<u>Handle account-based payment method credited notification from product perspective</u> / <u>Retrieve CV certificate over signing key</u> / <u>Retrieve CV certificate over signing key</u> / <u>Retrieve CV certificate over signing key</u>
<b>Linked Use Cases (Realises)</b>	<u>Asynchronous ION use case initiator side</u> / <u>Asynchronous ION use case initiator side</u>
<b>Base Activity</b>	
<b>Inputs</b>	<u>Terminal notify account-based payment method credited : tNotifyAccountBasedPaymentMethodCredited</u>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<u>CCP-BO-Main::Handle account-based payment method credited notification from operational perspective</u>
<b>Alternative 1</b>	
<b>Inputs</b>	<u>Terminal notify account-based payment method crediting aborted : tNotifyAccountBasedPaymentMethodCreditingAborted</u>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<u>CCP-BO-Main::Handle account-based payment method crediting aborted notification from operational perspective</u>

## 17.2.4 Handle account-based payment method credited notification from contractual perspective



<b>Use Case</b>	<a href="#">Handle account-based payment method credited notification from contractual perspective</a>
<b>Description</b>	<p>Handle an account-based payment method credited notification from the contractual perspective.</p> <p>The pCCP does its contractual checks and monitoring.</p> <p><b>Note:</b> if the pCCP itself performed the credit action, this use case takes inside the use case <a href="#">Handle account-based payment method credited notification from operational perspective</a> and is not called by the PO.</p> <p>In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a>.</p>
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases</b>	<a href="#">Demand entitlement hotlisting</a> / <a href="#">Demand application hotlisting</a>



<b>(Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward account-based payment method credited notification : forwardAccountBasedPaymentMethodCreditedNotification</a>
<b>Outputs</b>	<a href="#">Forward account-based payment method credited notification response : forwardAccountBasedPaymentMethodCreditedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward account-based payment method credited notification exception : forwardAccountBasedPaymentMethodCreditedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle account-based payment method credited notification from contractual perspective</a>

# 18 Sale Electronic Ticket via Stored-Value Payment Bundle CCP-System

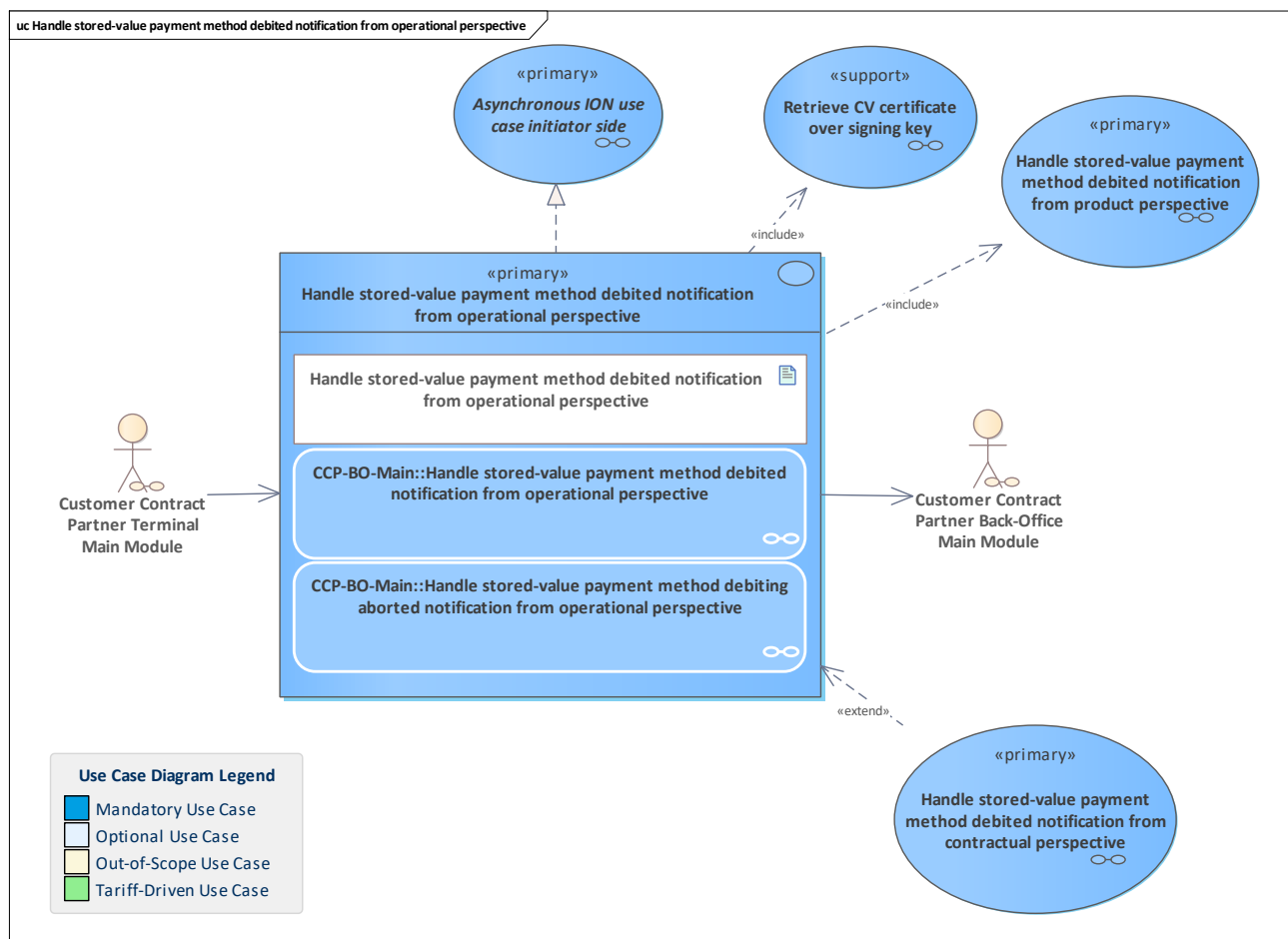
Functionality bundle that provides CCP back-office system use cases for selling or purchasing electronic tickets via a stored-value payment method on a user medium.

## 18.1 Overview

Handle stored-value payment method debited notification from operational perspective  
Handle stored-value payment method debited notification from contractual perspective  
Handle stored-value payment method credited notification from operational perspective  
Handle stored-value payment method credited notification from contractual perspective

## 18.2 Use Cases

### 18.2.1 Handle stored-value payment method debited notification from operational perspective

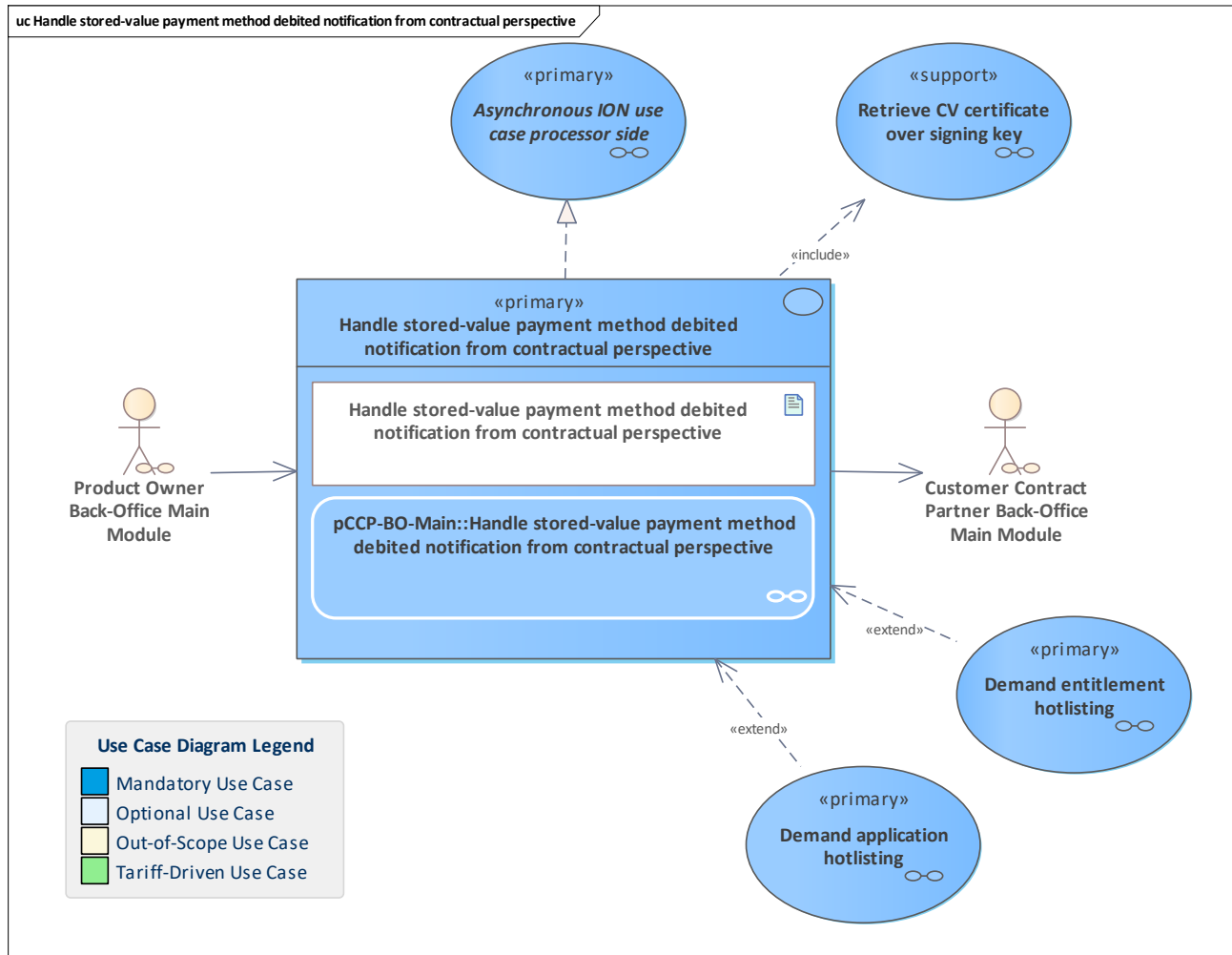




<b>Use Case</b>	<a href="#">Handle stored-value payment method debited notification from operational perspective</a>
<b>Description</b>	<p>Handle a notification about a stored-value payment method debiting from the operational perspective.</p> <p>The CCP back-office system receives the notification about the debit transaction of a stored-value payment method and registers it.</p> <p>Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the debit transaction attestation.</p> <p>Then, the notification is forwarded to the responsible PO system. If the current CCP is the pCCP, it can also do the contractual checks and monitoring directly.</p> <p>In case of a transaction abortion, the notification is registered for consistent monitoring. Since no counters are affected which are important to the PO, the notification is not forwarded.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle stored-value payment method debited notification from contractual perspective</a> / <a href="#">Handle stored-value payment method debited notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle stored-value payment method debited notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify stored-value payment method debited : tNotifyStoredValuePaymentMethodDebited</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle stored-value payment method debited notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify stored-value payment method debited aborted : tNotifyStoredValuePaymentMethodDebitingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle stored-value payment method debiting aborted notification from operational perspective</a>



## 18.2.2 Handle stored-value payment method debited notification from contractual perspective

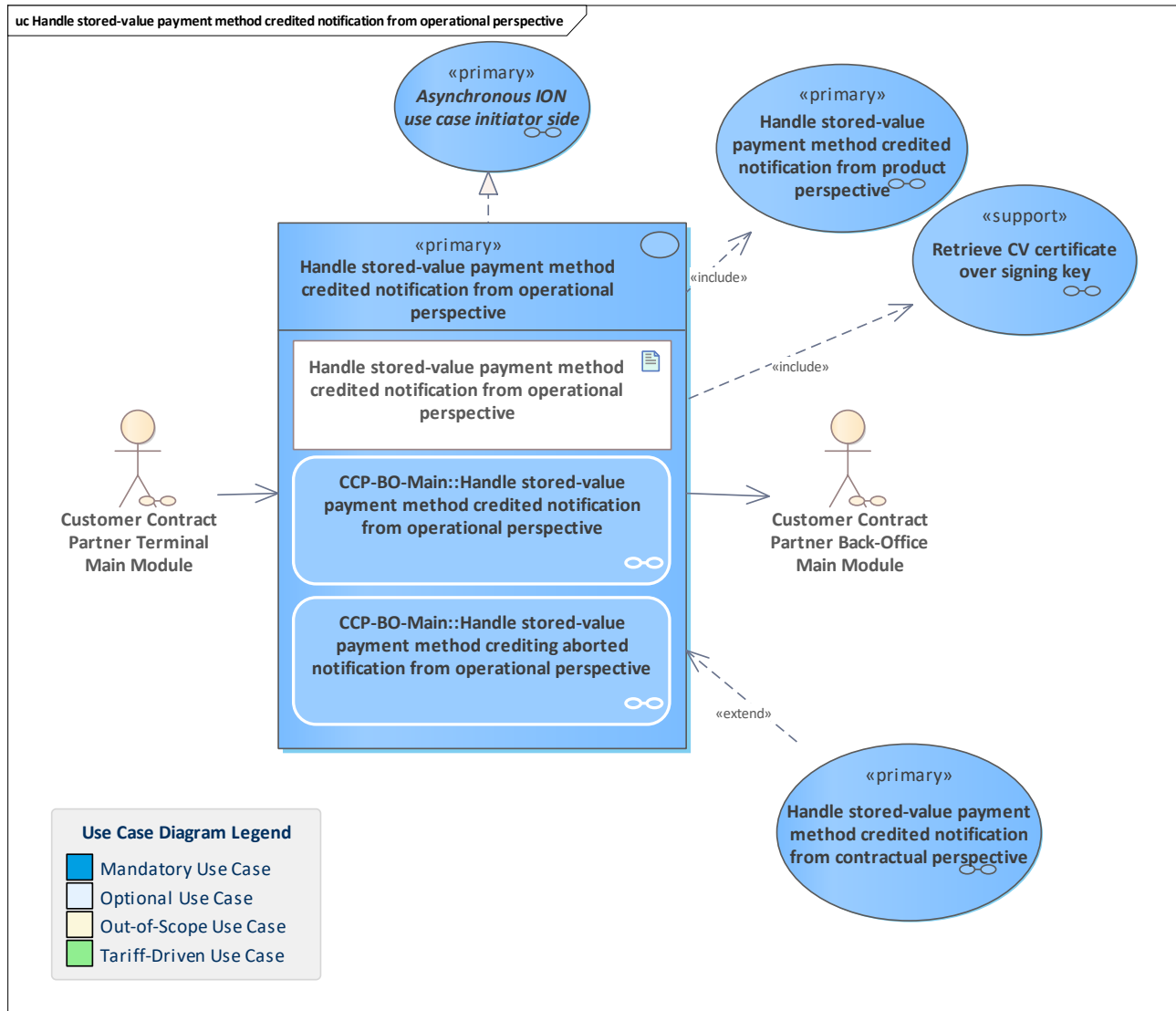


<b>Use Case</b>	<a href="#">Handle stored-value payment method debited notification from contractual perspective</a>
<b>Description</b>	<p>Handle a notification about a stored-value payment method debiting from the contractual perspective.</p> <p>The pCCP does its contractual checks and monitoring.</p> <p>Note: if the pCCP itself performed the debit action, this use case takes place inside the use case <a href="#">Handle stored-value payment method debited notification from operational perspective</a> and is not called by the PO.</p> <p>In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a>.</p>
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Demand entitlement hotlisting</a>
<b>Linked Use Cases</b>	<a href="#">Retrieve CV certificate over signing key</a>



<b>(Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward stored-value payment method debited notification : forwardStoredValuePaymentMethodDebitedNotification</a>
<b>Outputs</b>	<a href="#">Forward stored-value payment method debited notification response : forwardStoredValuePaymentMethodDebitedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward stored-value payment method debited notification exception : forwardStoredValuePaymentMethodDebitedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle stored-value payment method debited notification from contractual perspective</a>

## 18.2.3 Handle stored-value payment method credited notification from operational perspective

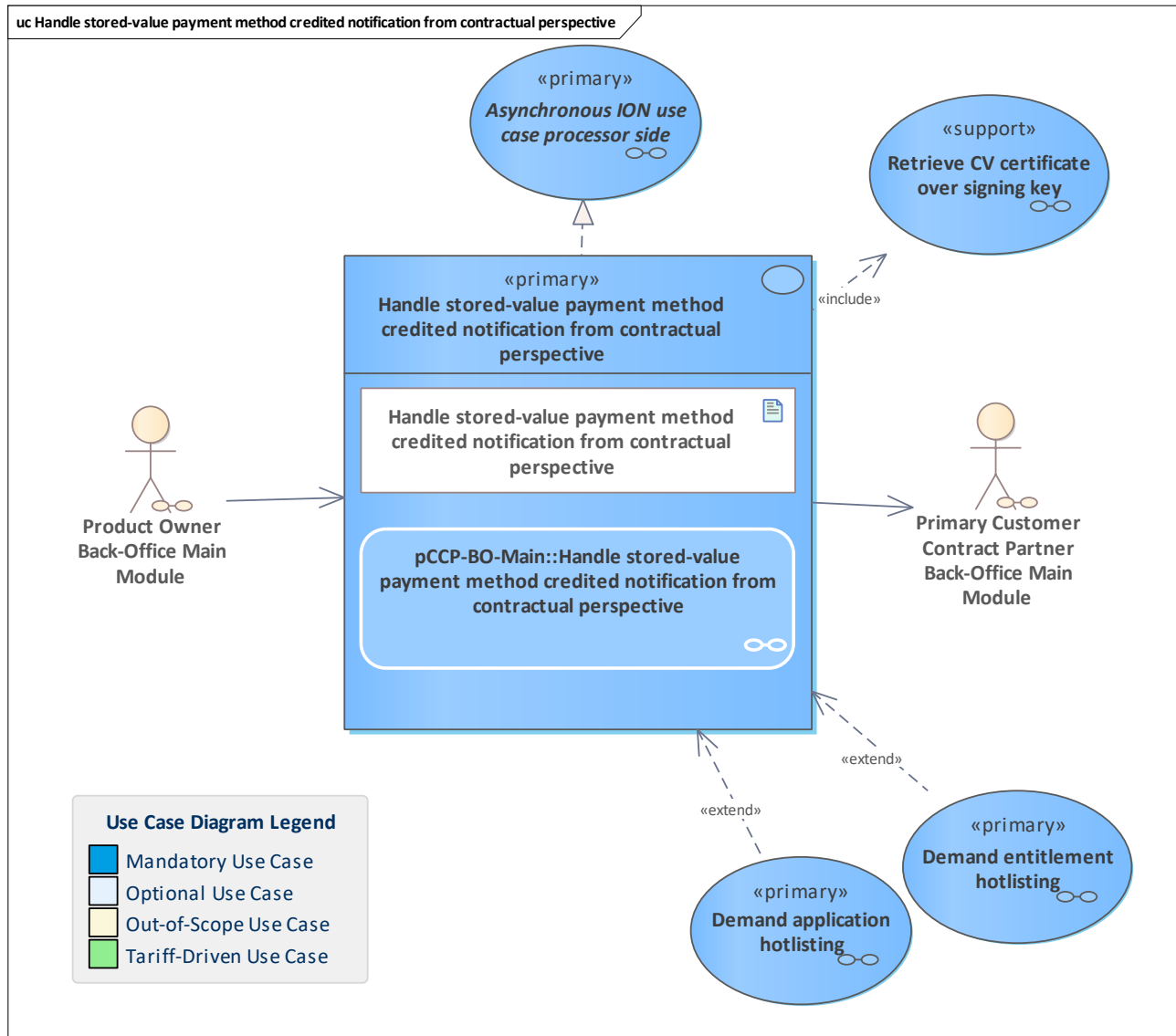


Use Case	<a href="#">Handle stored-value payment method credited notification from operational perspective</a>
Description	<p>Handle a stored-value payment method credited notification from the operational perspective.</p> <p>The CCP back-office system receives the notification about the credit transaction of a stored-value payment method and registers it.</p> <p>Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the credit action attestation.</p> <p>Then, the notification is forwarded to the responsible PO system.</p> <p>If the current CCP is the pCCP, it can also do the contractual checks and monitoring directly.</p> <p>In case of an abortion, the notification is registered for consistent monitoring. Since no counters are affected which are important to</p>



	the PO, in the case of a transaction abortion, the notification is not forwarded.
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle stored-value payment method credited notification from contractual perspective</a> / <a href="#">Handle stored-value payment method credited notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle stored-value payment method credited notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify stored-value payment method credited : tNotifyStoredValuePaymentMethodCredited</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle stored-value payment method credited notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify stored-value payment method crediting aborted : tNotifyStoredValuePaymentMethodCreditingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle stored-value payment method crediting aborted notification from operational perspective</a>

## 18.2.4 Handle stored-value payment method credited notification from contractual perspective



<b>Use Case</b>	<a href="#">Handle stored-value payment method credited notification from contractual perspective</a>
<b>Description</b>	<p>Handle a stored-value payment method credited notification from the contractual perspective.</p> <p>The pCCP does its contractual checks and monitoring.</p> <p><b>Note:</b> if the pCCP itself performed the credit action, this use case takes inside the use case <a href="#">Handle stored-value payment method credited notification from operational perspective</a> and is not called by the PO.</p> <p>In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a>.</p>
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	



<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Demand entitlement hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward stored-value payment method credited notification : forwardStoredValuePaymentMethodCreditedNotification</a>
<b>Outputs</b>	<a href="#">Forward stored-value payment method credited notification response : forwardStoredValuePaymentMethodCreditedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward stored-value payment method credited notification exception : forwardStoredValuePaymentMethodCreditedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle stored-value payment method credited notification from contractual perspective</a>



## 19 IN-OUT Bundle CCP-System

Functionality bundle that provides CCP back-office system use cases for IN-OUT functionality. The term CICO is also used.

### 19.1 Overview

Handle check-in notification from contractual perspective

Handle check-out notification from contractual perspective

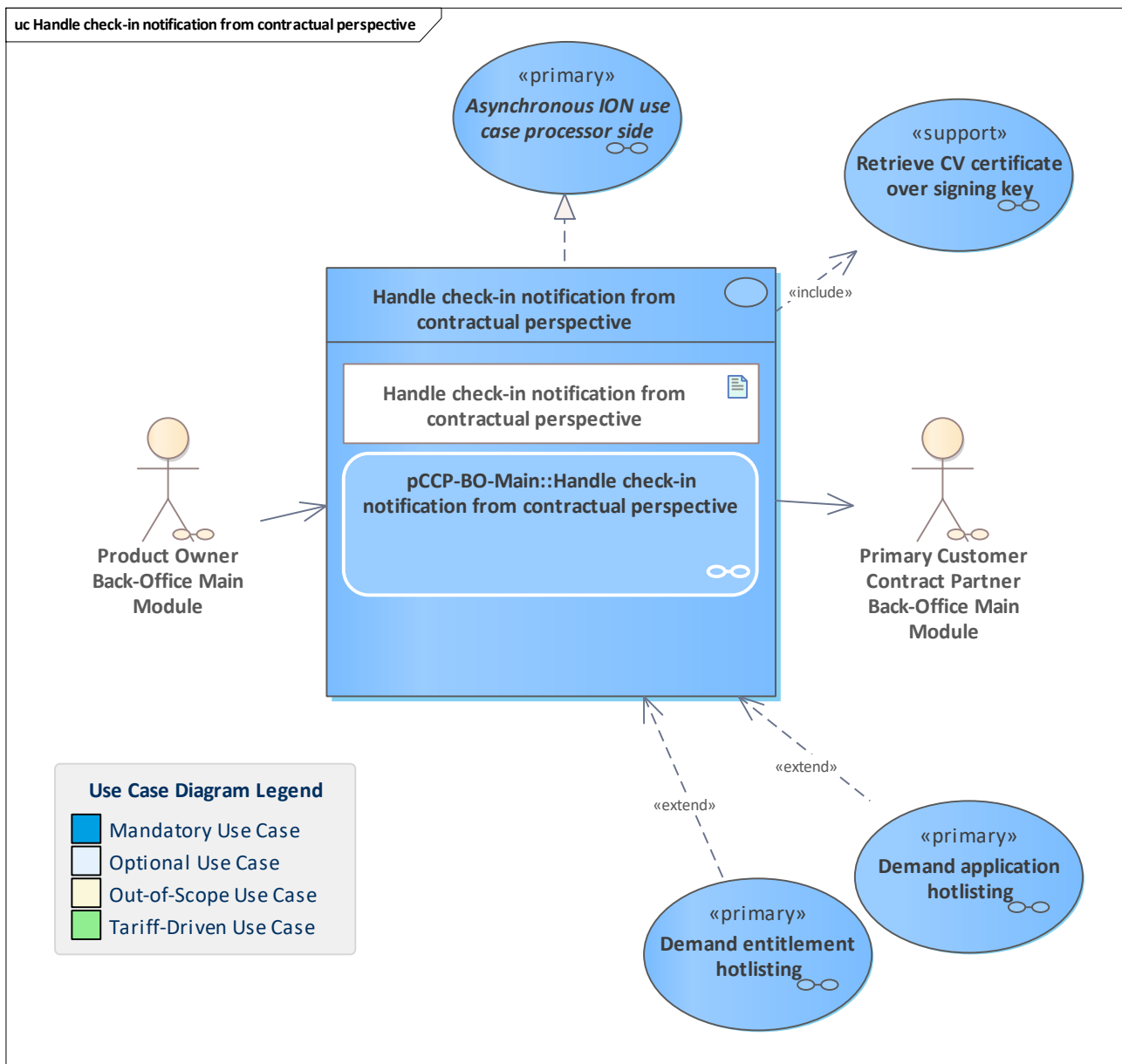
Optional: Handle user tariff parameters changed notification from operational perspective

Optional: Handle user tariff parameters changed notification from contractual perspective

Optional: Handle account-based payment method charging from contractual perspective

### 19.2 Use Cases

#### 19.2.1 Handle check-in notification from contractual perspective



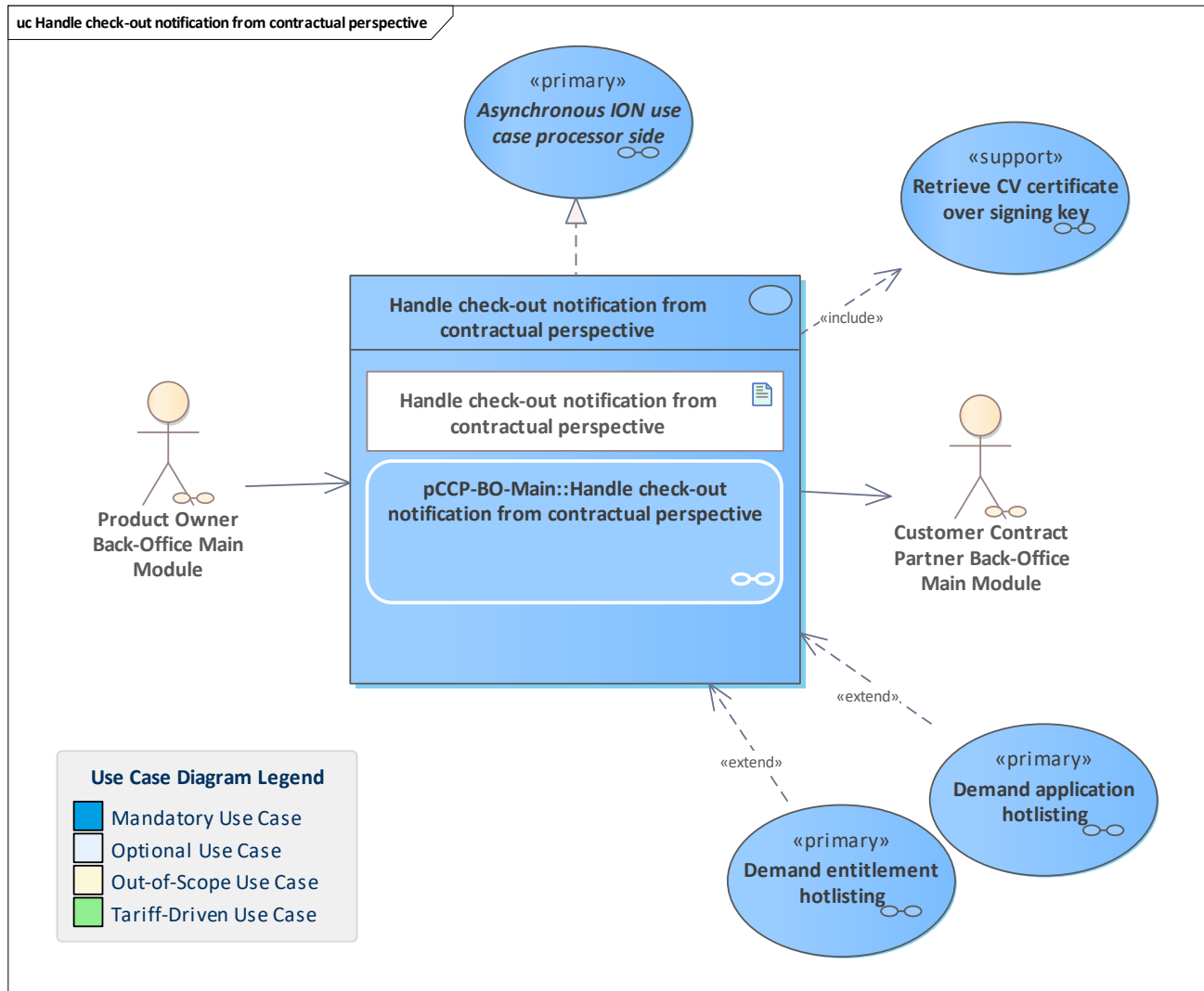
<b>Use Case</b>	<a href="#">Handle check-in notification from contractual perspective</a>
<b>Description</b>	Handle a check-in operation from the contractual perspective. The pCCP receives the notification from the PO system, registers it and does its contractual monitoring checks. For a potential later billing, this notification will be referenced by the PO system when sending the collected and rated recording events.
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Demand entitlement hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases</b>	<a href="#">Asynchronous ION use case processor side</a>





<b>(Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<u>Forward check in notification : forwardCheckinNotification</u>
<b>Outputs</b>	<u>Forward check in notification response :</u> <u>forwardCheckinNotificationResponse</u>
<b>Error Cases</b>	<u>E CCP RECEIVER IS NOT ENTITY OWNER</u> <u>E CO BINARY STRUCTURE CANNOT BE PROCESSED</u> <u>E CO WRONG ATTESTATION IN NOTIFICATION</u> <u>E CO WRONG SECURITY LEVEL</u> <u>E CO DUPLICATE UM ATTESTATION</u> <u>Forward check in notification exception :</u> <u>forwardCheckinNotificationException</u>
<b>Activity Diagram</b>	<u>pCCP-BO-Main::Handle check-in notification from contractual perspective</u>

## 19.2.2 Handle check-out notification from contractual perspective

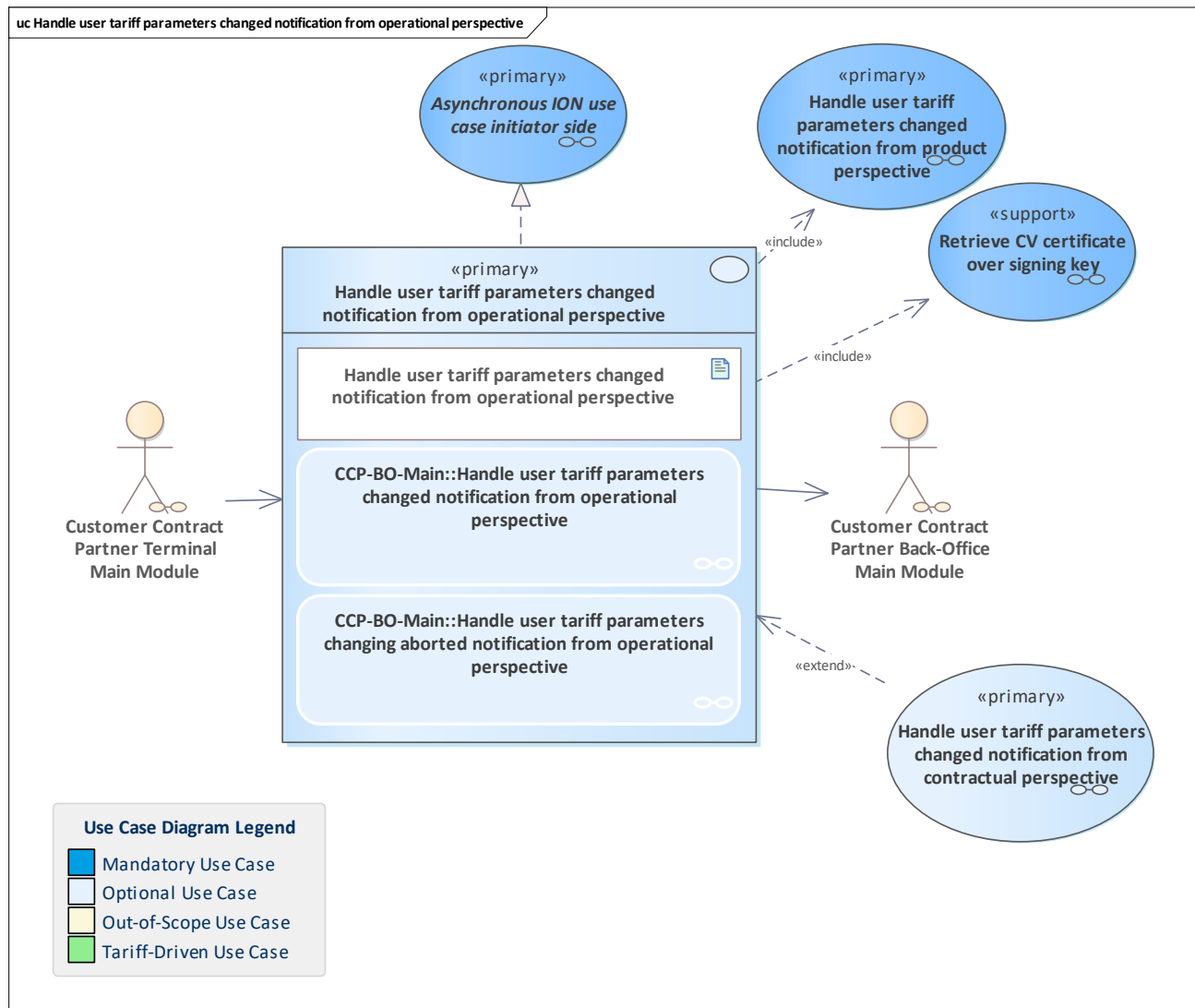


<b>Use Case</b>	<a href="#">Handle check-out notification from contractual perspective</a>
<b>Description</b>	Handle check-out operation from the contractual perspective. The pCCP receives the notification from the PO system, registers it and does its contractual monitoring checks. For a potential later billing, this notification will be referenced by the PO system when sending the collected and rated recording events.
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand entitlement hotlisting</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases</b>	<a href="#">Asynchronous ION use case processor side</a>



<b>(Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward check out notification : forwardCheckoutNotification</a>
<b>Outputs</b>	<a href="#">Forward check out notification response : forwardCheckoutNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward check out notification exception : forwardCheckoutNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle check-out notification from contractual perspective</a>

## 19.2.3 Optional: Handle user tariff parameters changed notification from operational perspective

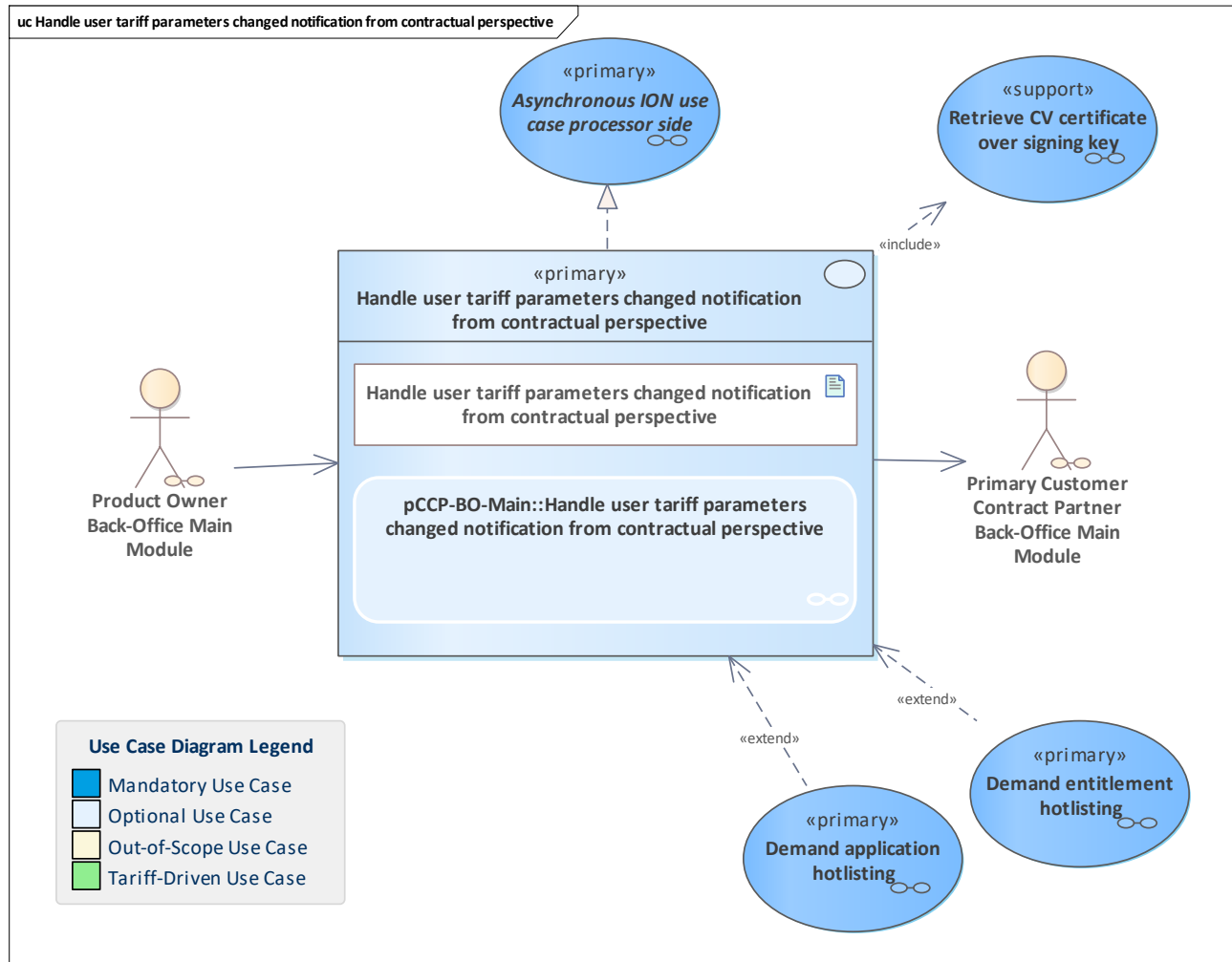


<b>Use Case</b>	<a href="#">Handle user tariff parameters changed notification from operational perspective</a>
<b>Description</b>	<p>Handle notification regarding changed user tariff parameters of an entitlement from the operational perspective.</p> <p>The changed user tariff parameters notification is sent by the CCP terminal to the CCP back-office system. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation.</p> <p><u>If the pCCP has changed user tariff parameters for its own entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO</li> <li>the pCCP does its contractual checks and monitoring</li> </ul> <p><u>If an sCCP has changed user tariff parameters for the entitlement:</u></p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO (and the PO will forward it</li> </ul>



	later to the pCCP)  In the case of action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The CCP back-office system registers the abortion for internal monitoring and data consistency.
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Handle user tariff parameters changed notification from contractual perspective</a> / <a href="#">Handle user tariff parameters changed notification from contractual perspective</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle user tariff parameters changed notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Handle user tariff parameters changed from terminal : tNotifyUserTariffParametersChanged</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle user tariff parameters changed notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Notify user tariff parameters changing aborted from terminal : tNotifyUserTariffParametersChangingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Handle user tariff parameters changing aborted notification from operational perspective</a>

## 19.2.4 Optional: Handle user tariff parameters changed notification from contractual perspective

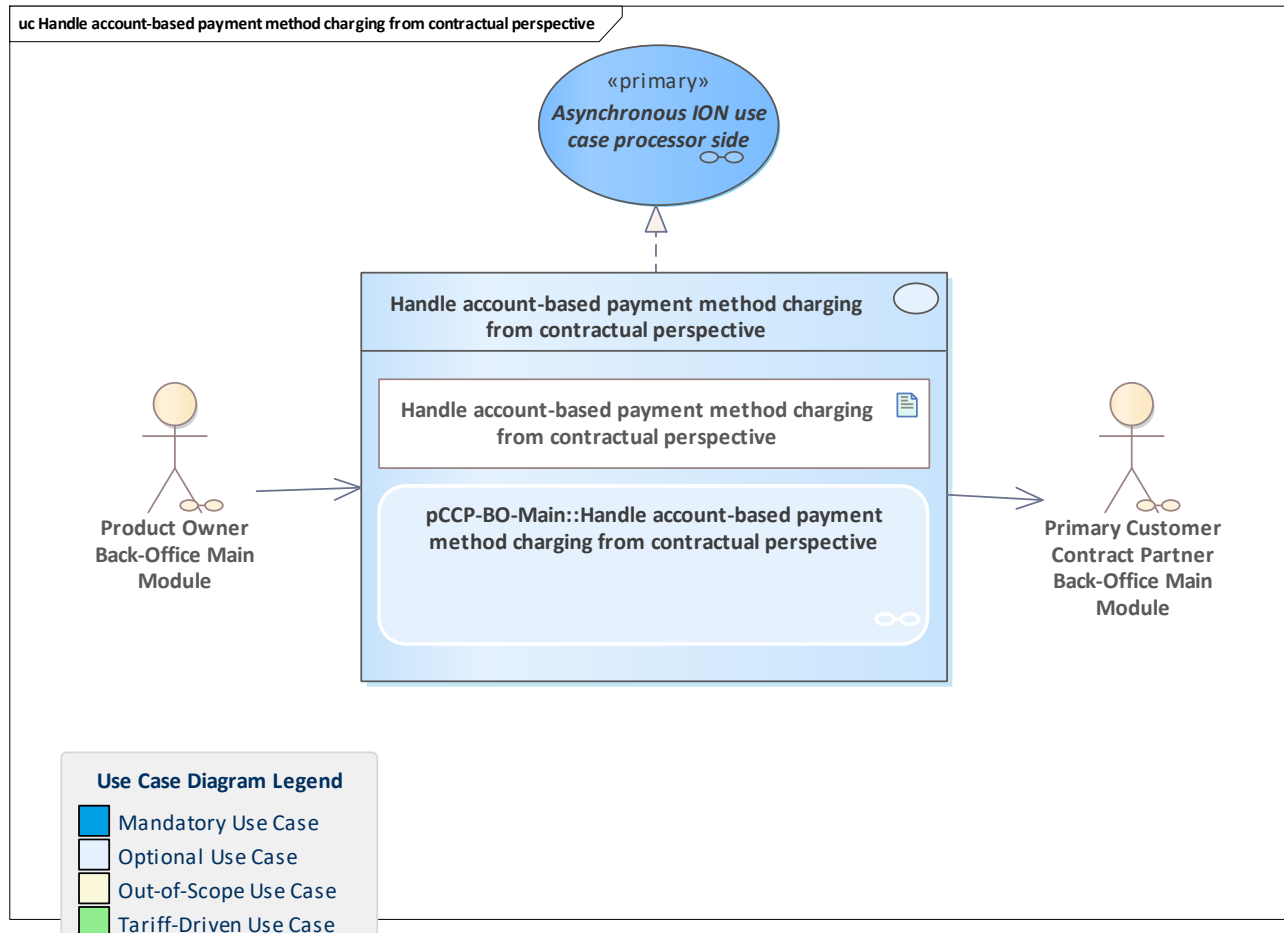


<b>Use Case</b>	<a href="#">Handle user tariff parameters changed notification from contractual perspective</a>
<b>Description</b>	<p>Handling a notification regarding changed user tariff parameters from the contractual perspective of the pCCP.</p> <p>The entitlement changed user tariff parameters notification is received by the pCCP.</p> <p>The pCCP does its checks and monitoring from the contractual perspective regarding the correct execution. In this context, the signature of the embedded attestation is verified.</p> <p><b>Note:</b> if the pCCP itself performed the changing of the user tariff parameters, this use case takes place inside the use case <a href="#">Handle user tariff parameters changed notification from operational perspective</a> and is not called by the PO. In this case, this use case is not an <a href="#">Asynchronous ION use case processor side</a>.</p>
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	



<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Demand entitlement hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward user tariff parameters changed notification : forwardUserTariffParametersChangedNotification</a>
<b>Outputs</b>	<a href="#">Forward user tariff parameters changed notification response : forwardUserTariffParametersChangedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward user tariff parameters changed notification exception : forwardUserTariffParametersChangedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle user tariff parameters changed notification from contractual perspective</a>

## 19.2.5 Optional: Handle account-based payment method charging from contractual perspective



<b>Use Case</b>	<a href="#">Handle account-based payment method charging from contractual perspective</a>
<b>Description</b>	<p>Account-based payment method charging is received from the PO and handled.</p> <p>The message contains a list of rated journeys with references to the messages about check-in and check-out procedures that had been sent before.</p> <p>The list has to be verified and the price must be booked from the related account-based payment method.</p> <p>Furthermore, the journey list can be used to provide an itemised bill e.g. via the web portal for the customer.</p>
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>





Base Activity	
Inputs	<a href="#">Charge account-based payment method : chargeAccountBasedPaymentMethod</a>
Outputs	<a href="#">Charge account-based payment response : chargeAccountBasedPaymentMethodResponse</a>
Error Cases	<a href="#">E CCP REFERENCED CICO NOTIFICATION NOT FOUND IN DAT ABASE</a> <a href="#">E CCP REFERENCED CICO NOT HAVE SAME ENTITLMENTID</a> <a href="#">E CCP REFERENCED CICO NOT HAVE SAME PRODUCTID</a> <a href="#">E CCP PRODUCTID OF REFERENCED CICO NOT ABPM</a> <a href="#">Charge account-based payment method exception : chargeAccountBasedPaymentMethodException</a>
Activity Diagram	<a href="#">pCCP-BO-Main::Handle account-based payment method charging from contractual perspective</a>



## 20 Ordered Action Management Bundle Ordering-CCP-System

Functionality bundle that covers the use cases to implement the CCP back-office system functionality for working with remote action lists. This could include ordering or cancelling actions, for example.

### 20.1 Overview

[Order entitlement issuance](#)

[Order entitlement unblocking](#)

[Order entitlement termination](#)

[Order entitlement blocking](#)

[Order group](#)

[Cancel order](#)

[Handle order obsolescence notification](#)

[Handle ordered entitlement issued notification from contractual perspective](#)

[Handle ordered entitlement unblocked notification from contractual perspective](#)

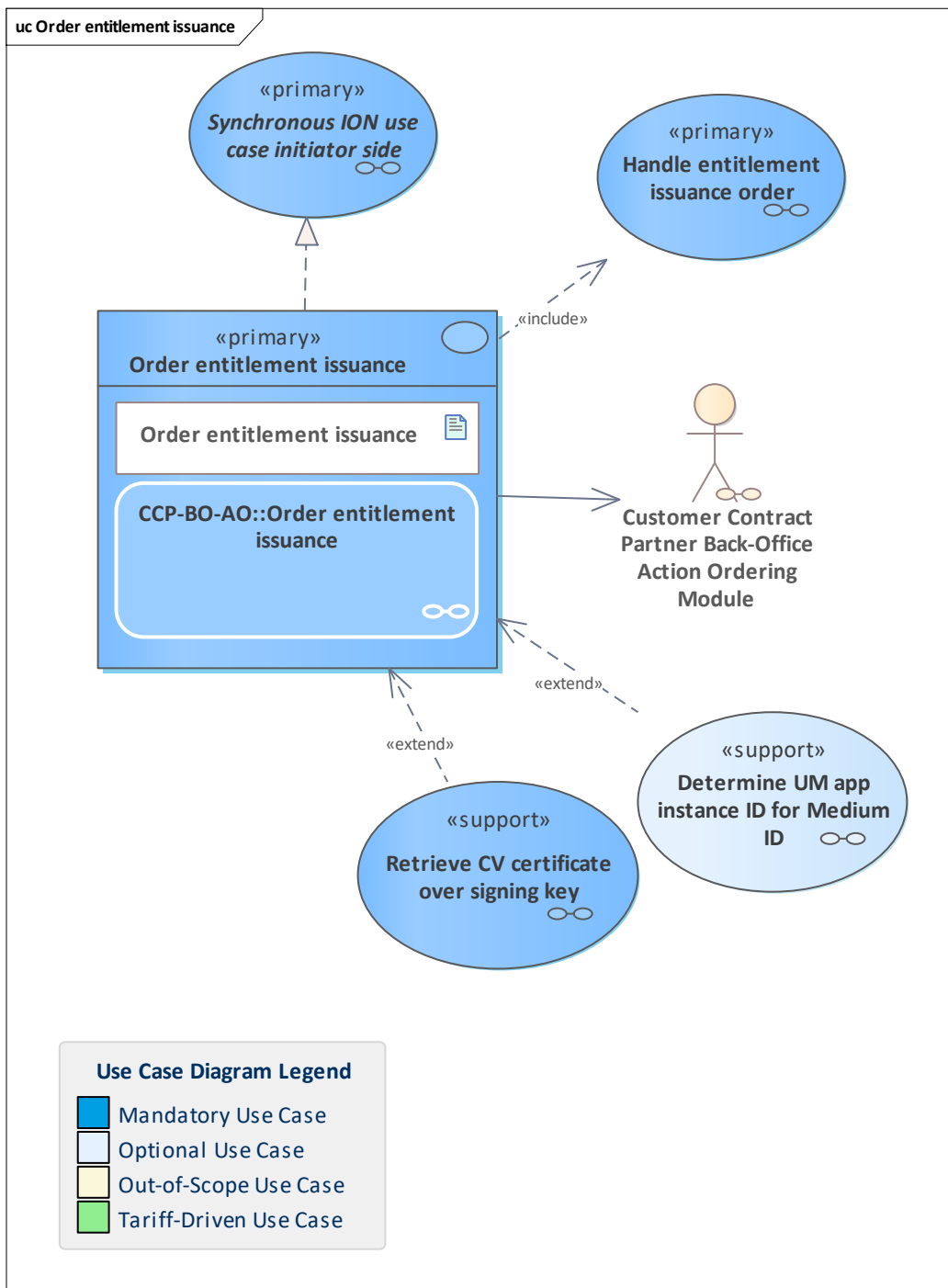
[Handle ordered entitlement terminated notification from contractual perspective](#)

[Handle ordered entitlement blocked notification from contractual perspective](#)

[Analyse order history from contractual perspective](#)

### 20.2 Use Cases

#### 20.2.1 Order entitlement issuance

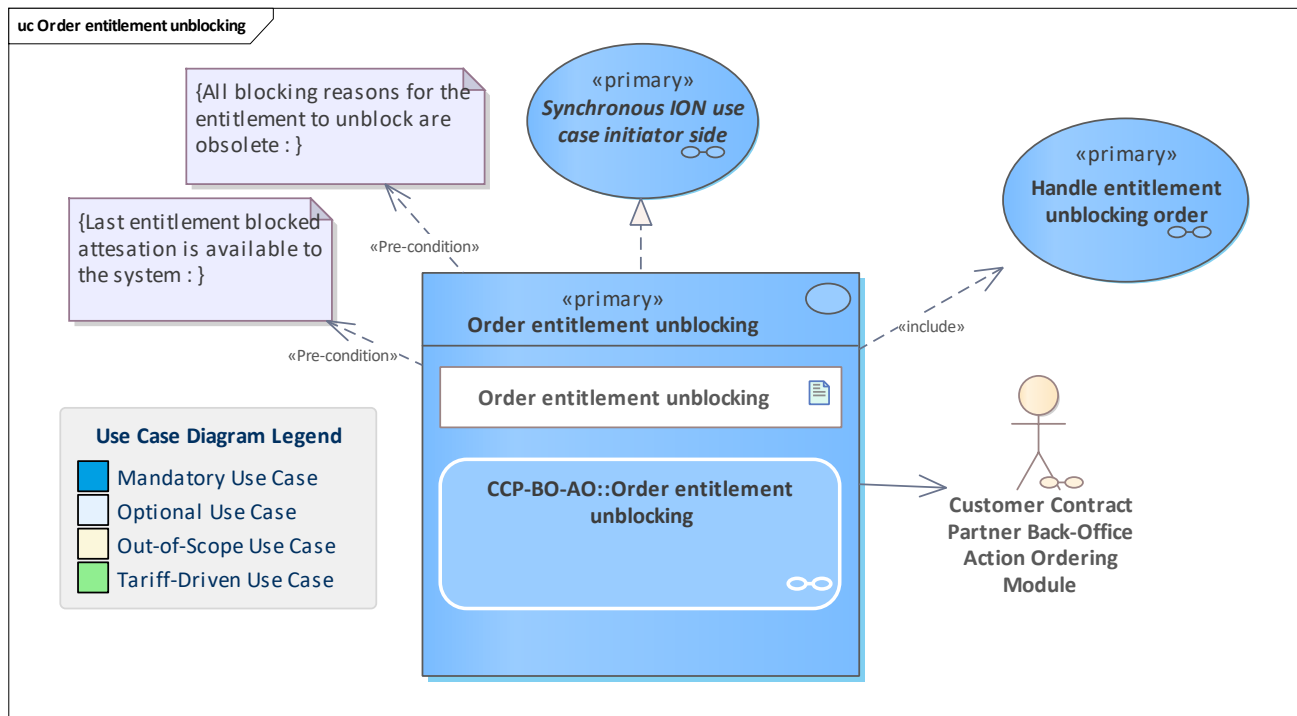


<b>Use Case</b>	<a href="#">Order entitlement issuance</a>
<b>Description</b>	The ordering CCP orders an entitlement issuance using ordered action execution.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Determine UM app instance ID for Medium ID</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases</b>	<a href="#">Handle entitlement issuance order</a> / <a href="#">Retrieve CV certificate over</a>



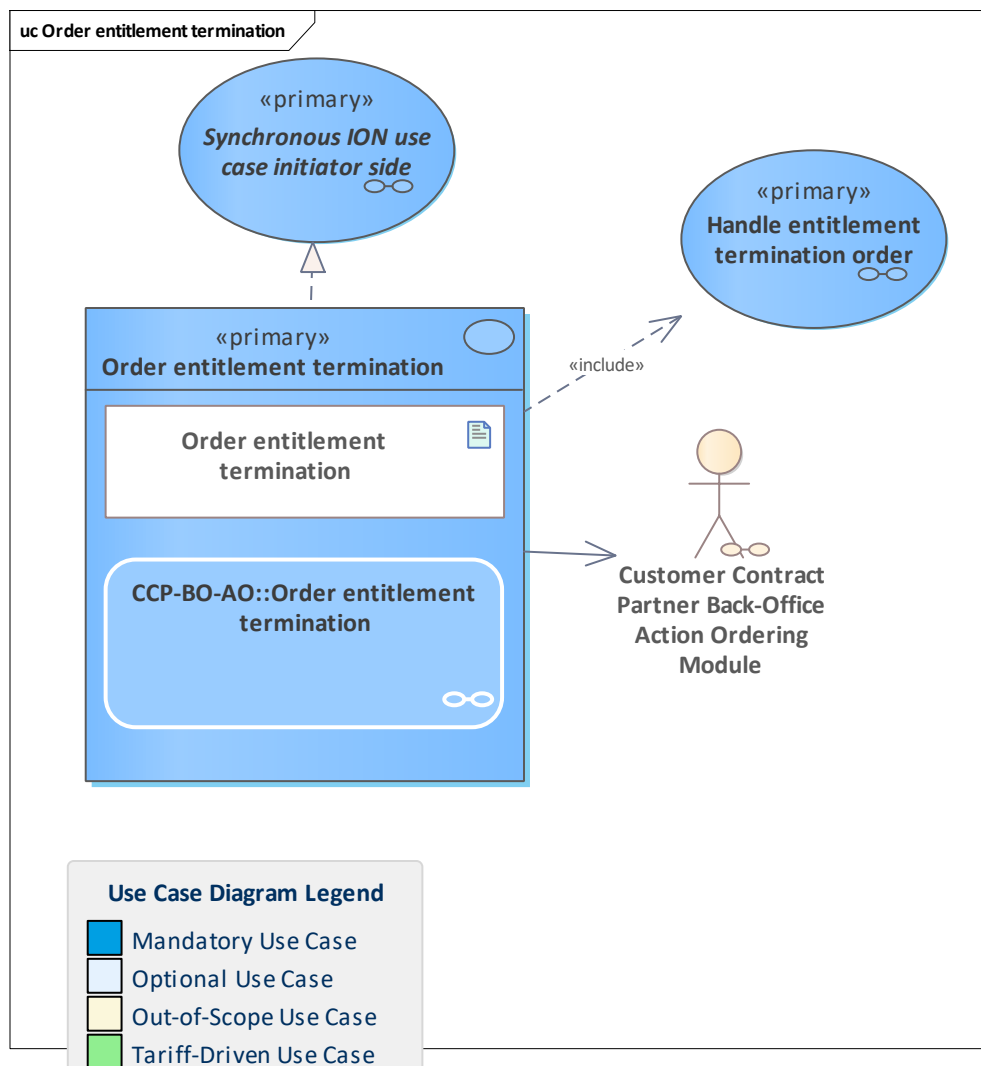
<b>(Includes)</b>	<a href="#">signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a> / <a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Action operator : OrganisationId</a> <a href="#">Entitlement effective time : EntitlementEffectiveTime</a> <a href="#">Entitlement expiration time : EntitlementExpirationTime</a> <a href="#">Entitlement issued metadata : EntitlementIssuedMetadata</a> <a href="#">Infotext : Infotext</a> <a href="#">Product ID : ProductId</a> <a href="#">Product parameters : ProductParameters</a> <a href="#">Stored-value autoload parameters : StoredValueAutoloadParameters</a> <a href="#">Stored-value parameters : StoredValueParameters</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Order entitlement issuance</a>

## 20.2.2 Order entitlement unblocking



<b>Use Case</b>	<a href="#">Order entitlement unblocking</a>
<b>Description</b>	The ordering CCP orders an entitlement unblocking. The unblocking may only be ordered after the corresponding blocked notification (resulting from a previous order) has been handled.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	<a href="#">All blocking reasons for the entitlement to unblock are obsolete</a> <a href="#">Last entitlement blocked attestation is available to the system</a>
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement unblocking order</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Action operator : OrganisationId</a> <a href="#">Signed entitlement blocked attestation : SignedEntitlementBlockedAttestation</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Order entitlement unblocking</a>

## 20.2.3 Order entitlement termination

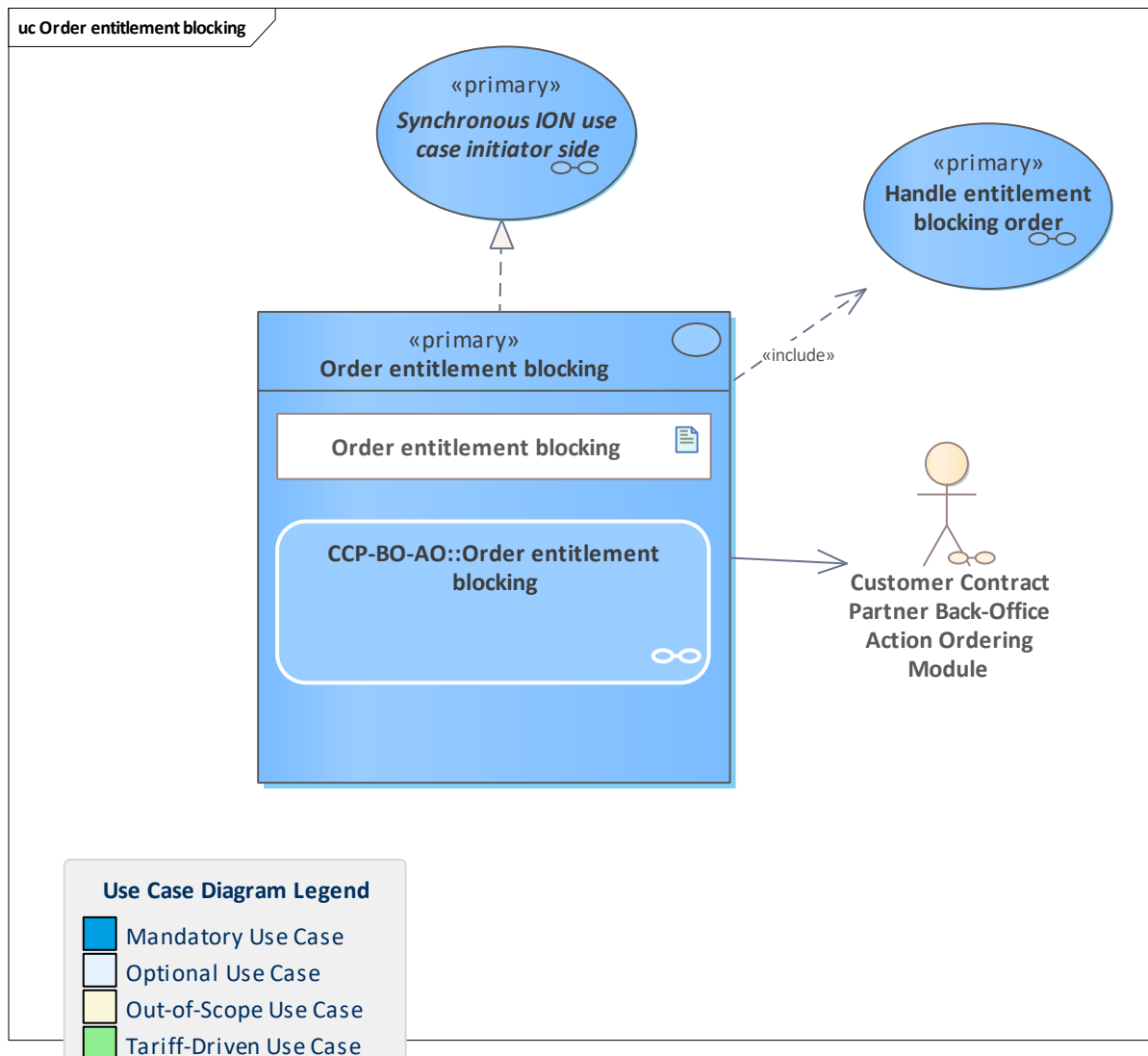


<b>Use Case</b>	<a href="#">Order entitlement termination</a>
<b>Description</b>	The ordering CCP orders an entitlement termination. The termination may only be ordered after the corresponding issuance notification (resulting from a previous order) has been handled.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement termination order</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Action operator : OrganisationId</a> <a href="#">Signed entitlement issued attestation :</a>



	<a href="#">SignedEntitlementIssuedAttestation</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Order entitlement termination</a>

## 20.2.4 Order entitlement blocking



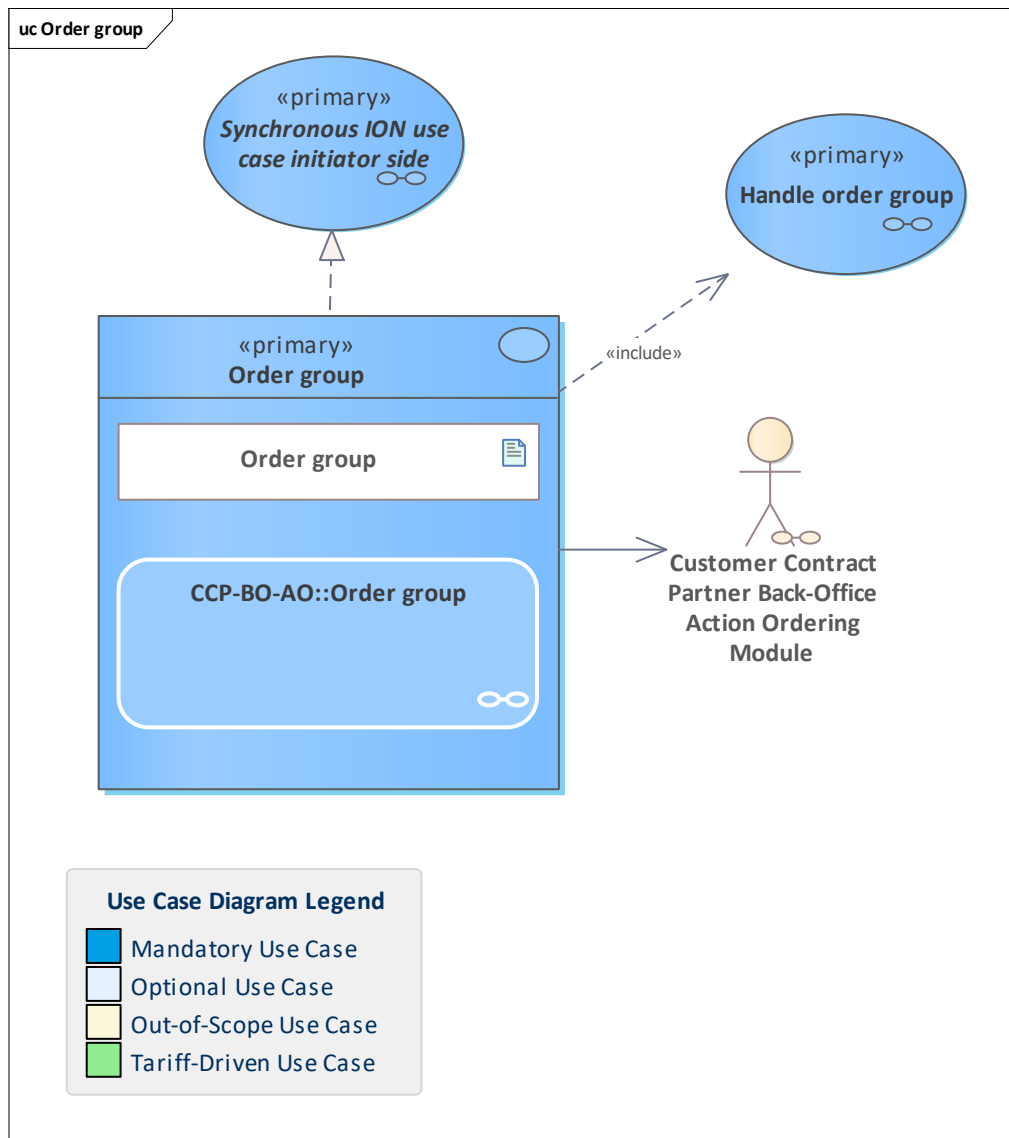
<b>Use Case</b>	<a href="#">Order entitlement blocking</a>
<b>Description</b>	<p>Using ordered action execution, the ordering CCP orders the blocking of an entitlement that might already have been issued, but whose issuance notification has not yet reached its owner system. Therefore, the blocking order is based on the order ID of the issuance order, since the entitlement ID is not available/known yet.</p> <p>As soon as the ordered entitlement issued notification reaches the ordering system, the blocking order should be cancelled (if still active) and replaced by a regular hotlist entry.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases</b>	<a href="#">Handle entitlement blocking order</a>





<b>(Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Action operator : OrganisationId</a> <a href="#">Entitlement issuance order to block the potential result of : EntitlementIssuanceOrder</a> <a href="#">Order expiration time : ZonedDateTime</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Order entitlement blocking</a>

## 20.2.5 Order group

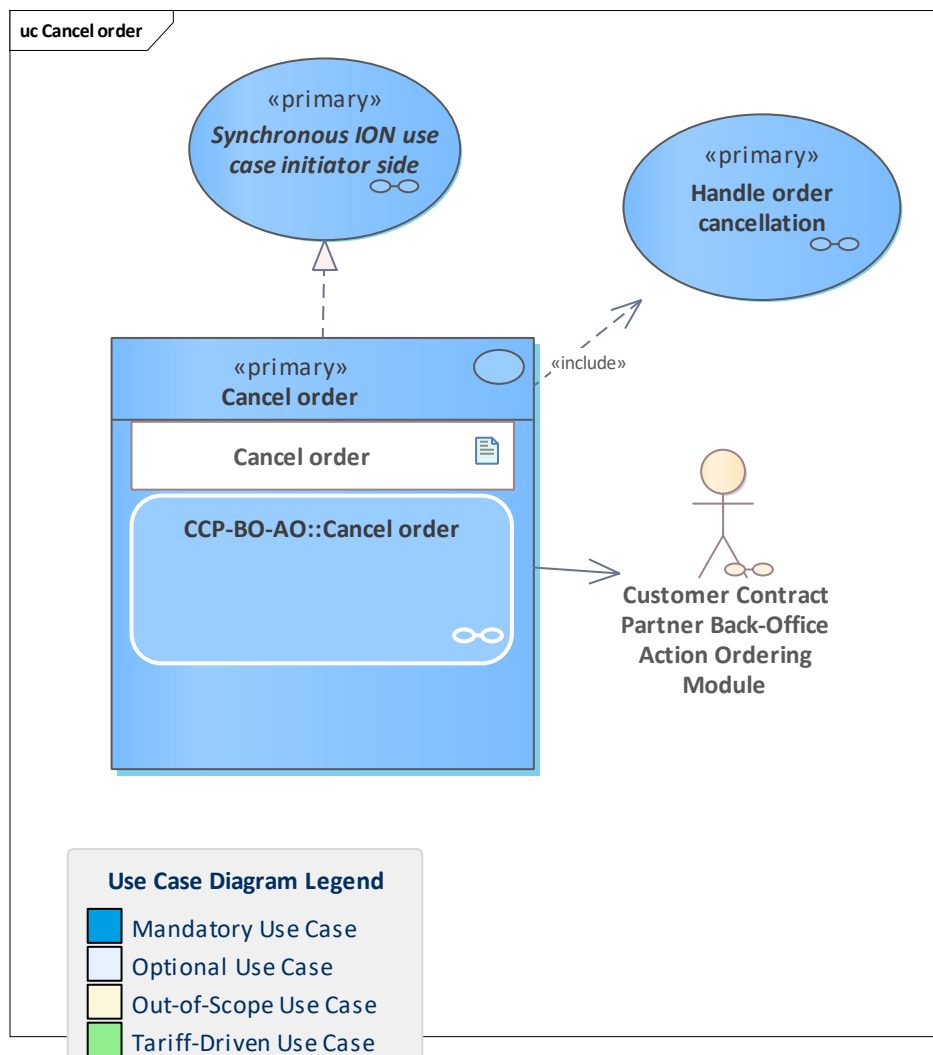


Use Case	<u>Order group</u>
Description	<p>The ordering CCP sends a set of orders to the <a href="#">Product Owner Back-Office Action Management Module</a>, which in turn will either accept all of them or reject all of them.</p> <p>The orders contained in the set have to target the same UM app instance ID and must share the same group ID.</p> <p>This process can be used, e.g., to change the product parameters of an entitlement at a future date X. For that purpose, the following orders would be created:</p> <ul style="list-style-type: none"> <li>• A termination order for the current entitlement</li> <li>• An issuance order for an entitlement identical to the current one, but with an expiration time of X</li> <li>• An issuance order for an entitlement identical to the current one, but with adjusted product parameters and effective time X</li> </ul>



	For all of these orders, the group ID would be set to the same, unique value, thus tying them together. Using the 'Order group' use case guarantees the atomicity of the change, thus ensuring that the customer is not left without an entitlement.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle order group</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Entitlement blocking orders : EntitlementBlockingOrder</a> <a href="#">Entitlement issuance orders : EntitlementIssuanceOrder</a> <a href="#">Entitlement termination orders : EntitlementTerminationOrder</a> <a href="#">Entitlement unblocking orders : EntitlementUnblockingOrder</a> <a href="#">Order cancellations : OrderId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Order_group</a>

## 20.2.6 Cancel order

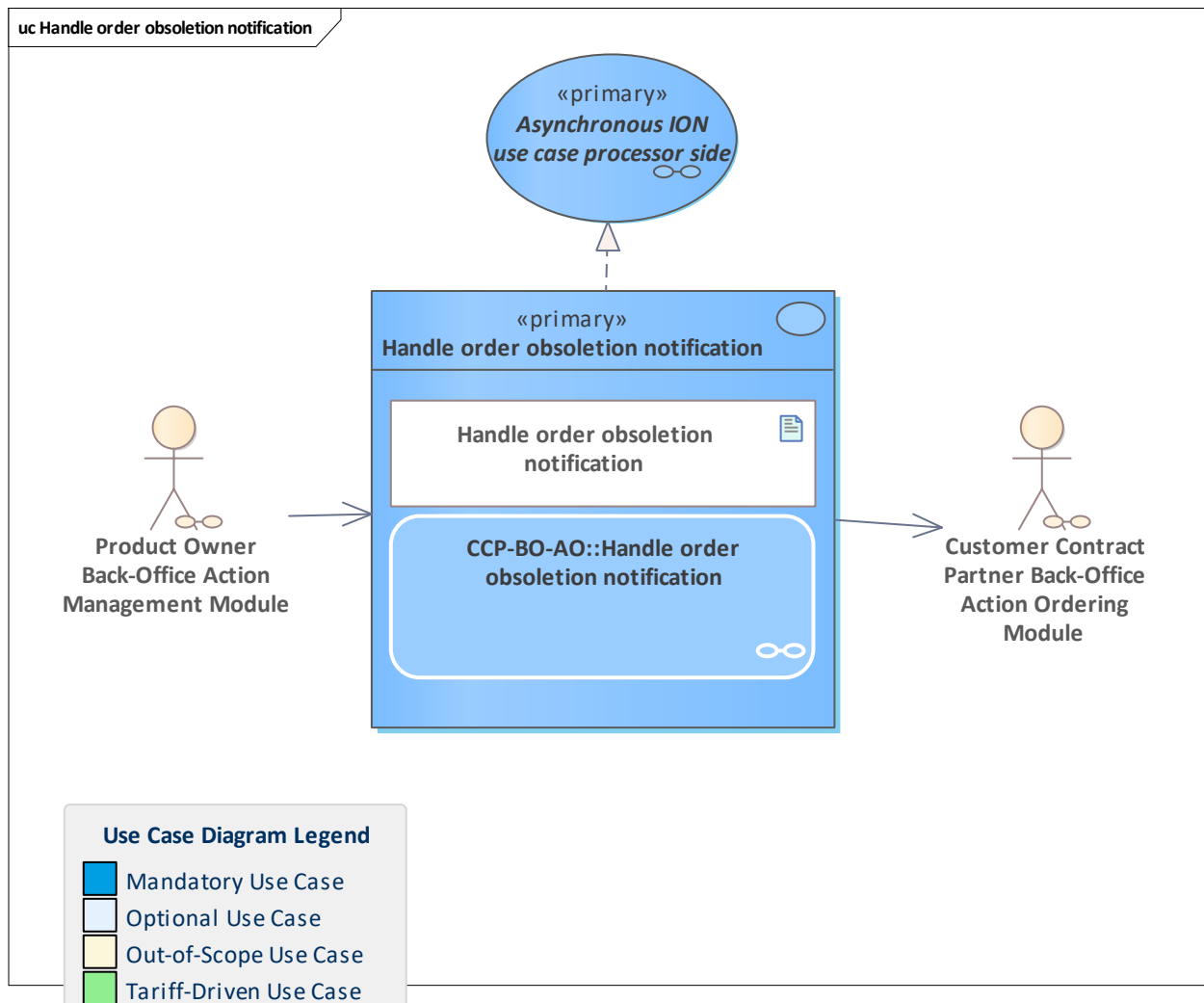


<b>Use Case</b>	<a href="#">Cancel order</a>
<b>Description</b>	The ordering CCP cancels an order. Cancelling an order will remove it from future action lists, but is not able to prevent the order from being executed before the next action list has reached all relevant terminals. In particular, the order may already have been executed and the execution notification may still be on its way to the relevant systems.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle order cancellation</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	



<b>Inputs</b>	<a href="#">Order ID : OrderId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Cancel order</a>

## 20.2.7 Handle order obsolescence notification

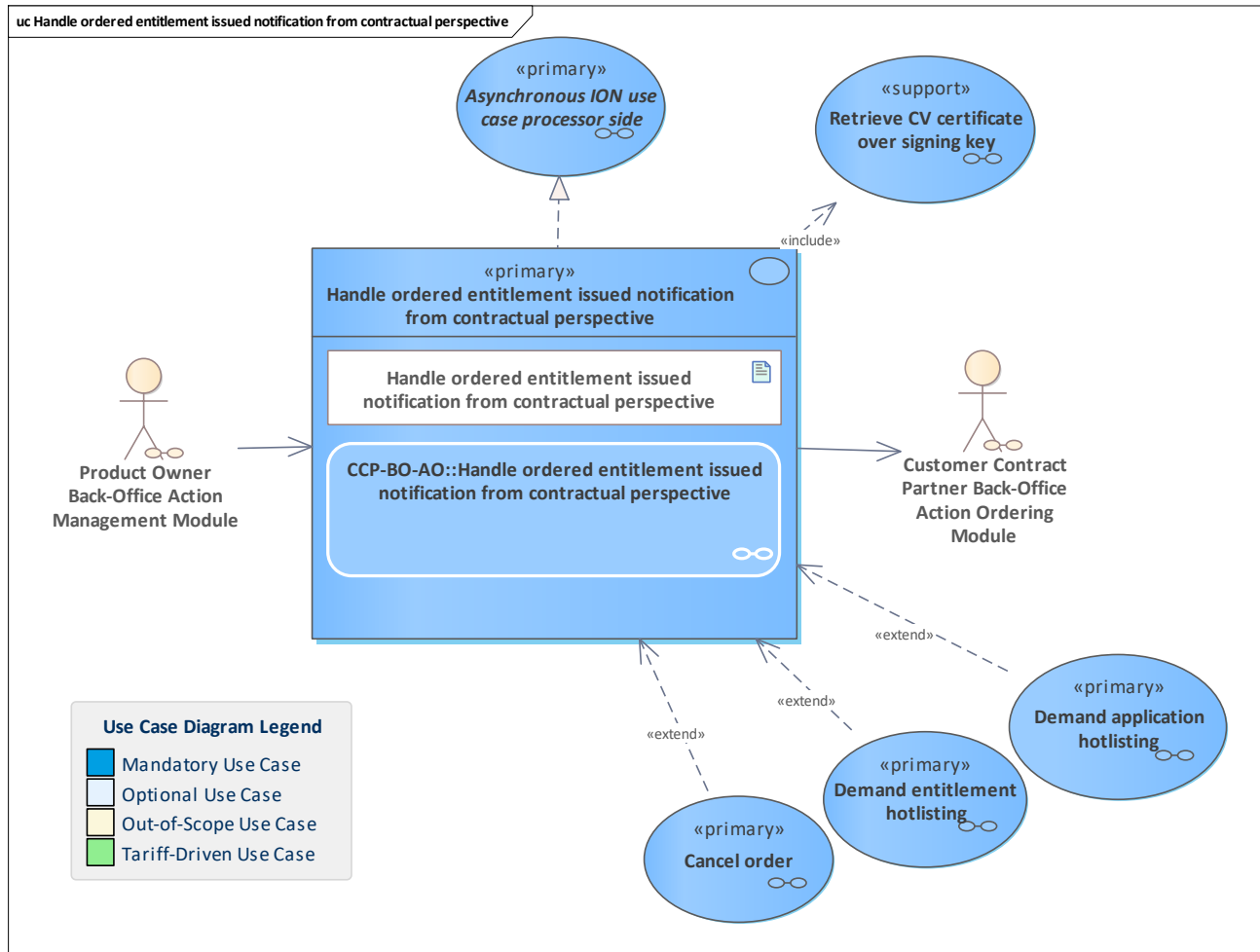


<b>Use Case</b>	<a href="#">Handle order obsolescence notification</a>
<b>Description</b>	Handle an order obsolescence (see <a href="#">Obsolete</a> ) notification triggered by action list clearing (see <a href="#">Check for order obsolescence</a> ).
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Action Management Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notify order seems to be obsolete :</a> <a href="#">notifyOrderSeemsToBeObsolete</a>
<b>Outputs</b>	<a href="#">Notify order seems to be obsolete response :</a> <a href="#">notifyOrderSeemsToBeObsoleteResponse</a>



<b>Error Cases</b>	<a href="#">E_CCPOAO_REFERENCED_ACTION_ORDER_UNKNOWN</a> <a href="#">Notify order seems to be obsolete exception :</a> <a href="#">notifyOrderSeemsToBeObsoleteException</a>
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Handle order obsoletion notification</a>

## 20.2.8 Handle ordered entitlement issued notification from contractual perspective



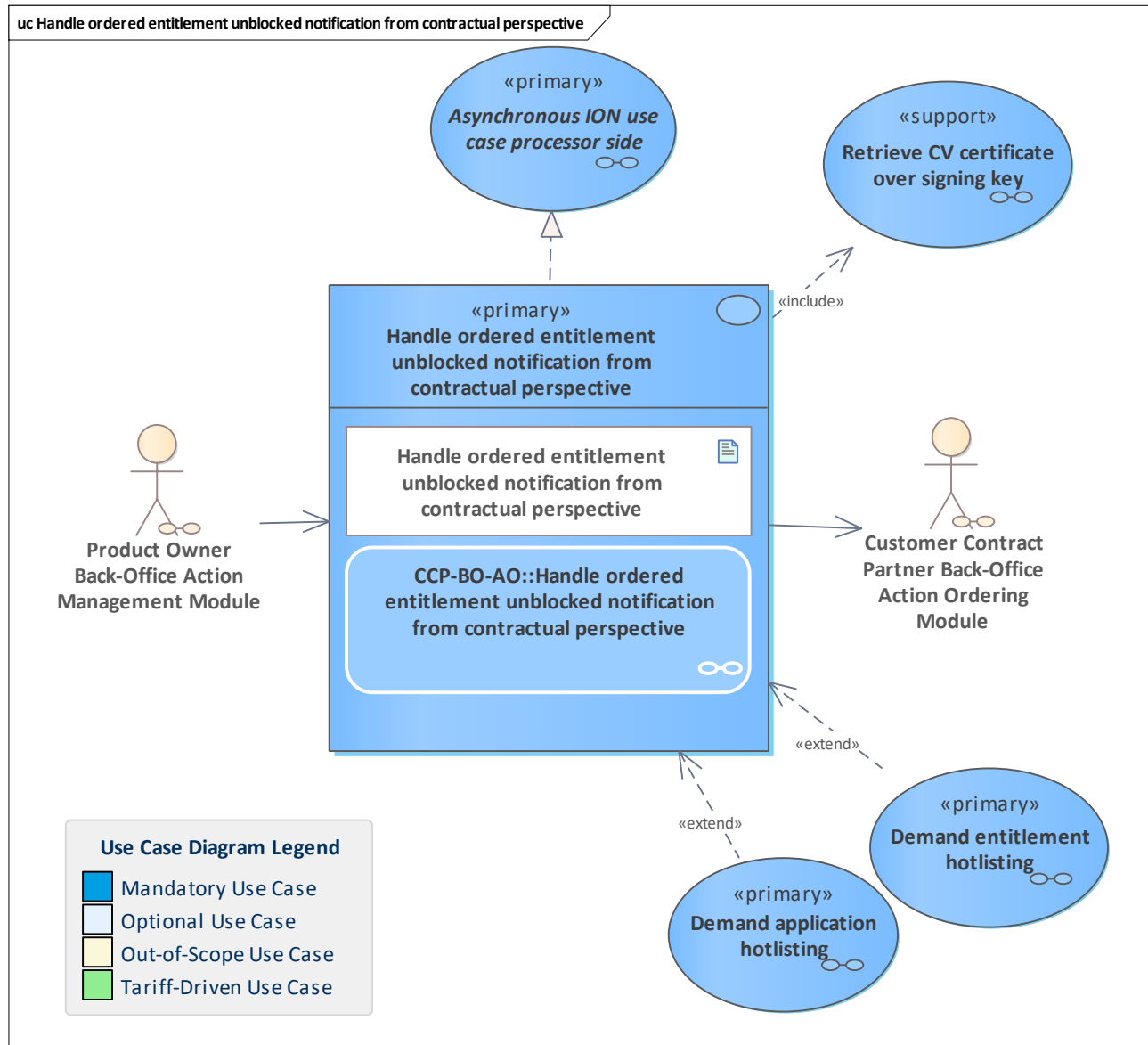
<b>Use Case</b>	<a href="#">Handle ordered entitlement issued notification from contractual perspective</a>
<b>Description</b>	<p>Handle the notification about a successful execution of an entitlement issuance ordered via action management from the contractual perspective.</p> <p>After the monitoring, the PO forwards the notification from the executing CCP to the ordering CCP. Thus, the <i>ordered entitlement issuance notification</i> is received by the CCP that initiated the ordered action.</p> <p>The CCP does its checks and monitoring from the contractual perspective regarding the correct execution of the ordered entitlement issuance. In this context, the signature of the embedded attestation is verified.</p> <p>Note that the application instance ID of the user medium the entitlement was issued to can be uniquely identified via <i>umAppInstanceId</i>, which is part of the <a href="#">SignedEntitlementIssuedAttestation</a> that is contained in the <a href="#">OrderedEntitlementIssuedNotification</a>.</p>
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Action Management Module</a>





<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Cancel order</a> / <a href="#">Demand entitlement hotlisting</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward ordered entitlement issued notification : forwardOrderedEntitlementIssuedNotification</a>
<b>Outputs</b>	<a href="#">Forward ordered entitlement issued notification response : forwardOrderedEntitlementIssuedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward ordered entitlement issued notification exception : forwardOrderedEntitlementIssuedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Handle ordered entitlement issued notification from contractual perspective</a>

## 20.2.9 Handle ordered entitlement unblocked notification from contractual perspective

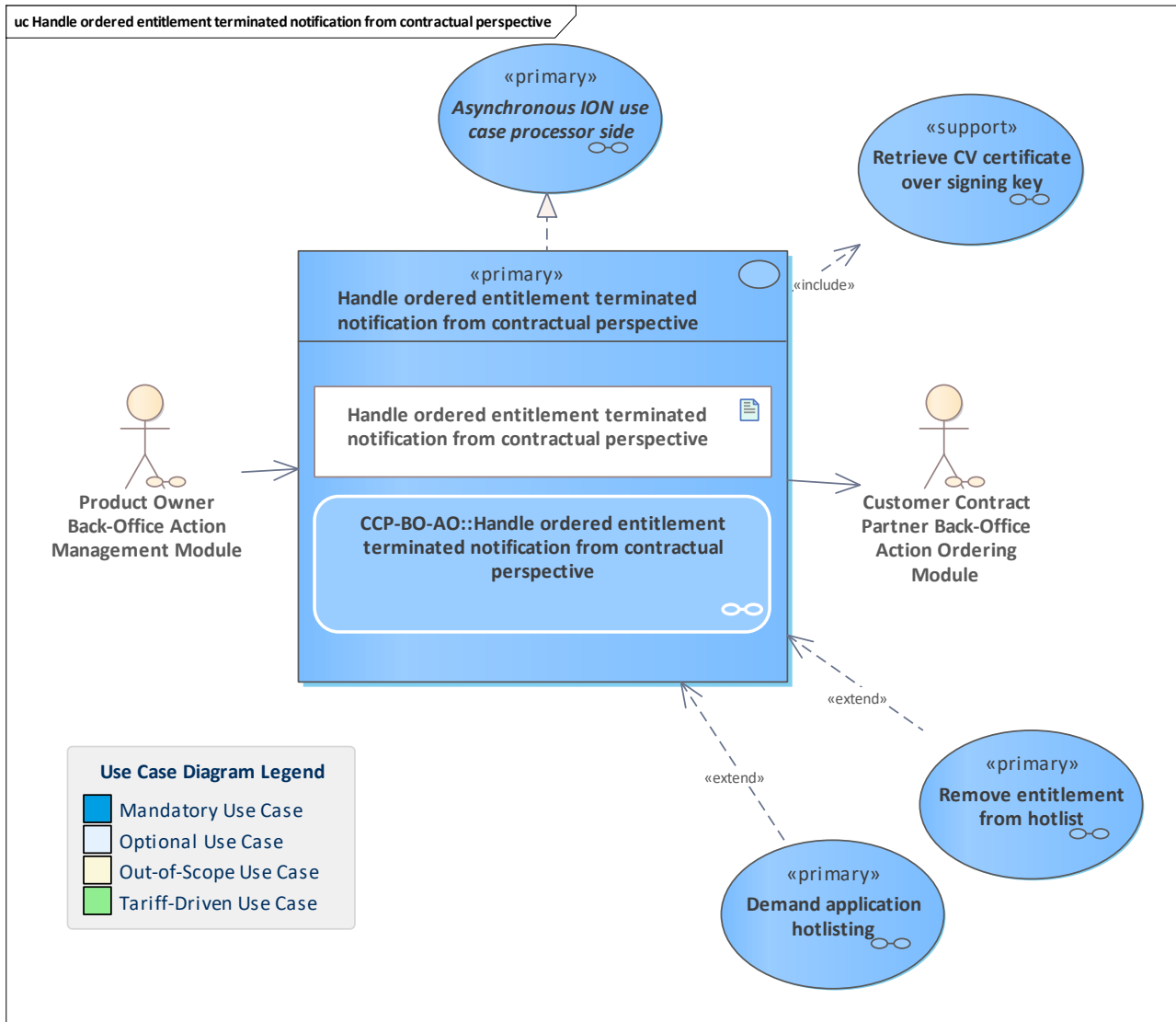


<b>Use Case</b>	<a href="#">Handle ordered entitlement unblocked notification from contractual perspective</a>
<b>Description</b>	<p>Handle the notification about a successful execution of an entitlement unblocking ordered via action management from the contractual perspective.</p> <p>The ordered entitlement unblocked notification is received by the ordering CCP.</p> <p>The CCP does its checks and monitoring from the contractual perspective regarding the correct execution of the unblocking. In this context, the signature of the embedded attestation is verified.</p>
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Action Management Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	



<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Demand entitlement hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward ordered entitlement unblocked notification : forwardOrderedEntitlementUnblockedNotification</a>
<b>Outputs</b>	<a href="#">Forward ordered entitlement unblocked notification response : forwardOrderedEntitlementUnblockedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward ordered entitlement unblocked notification exception : forwardOrderedEntitlementUnblockedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Handle ordered entitlement unblocked notification from contractual perspective</a>

## 20.2.10 Handle ordered entitlement terminated notification from contractual perspective

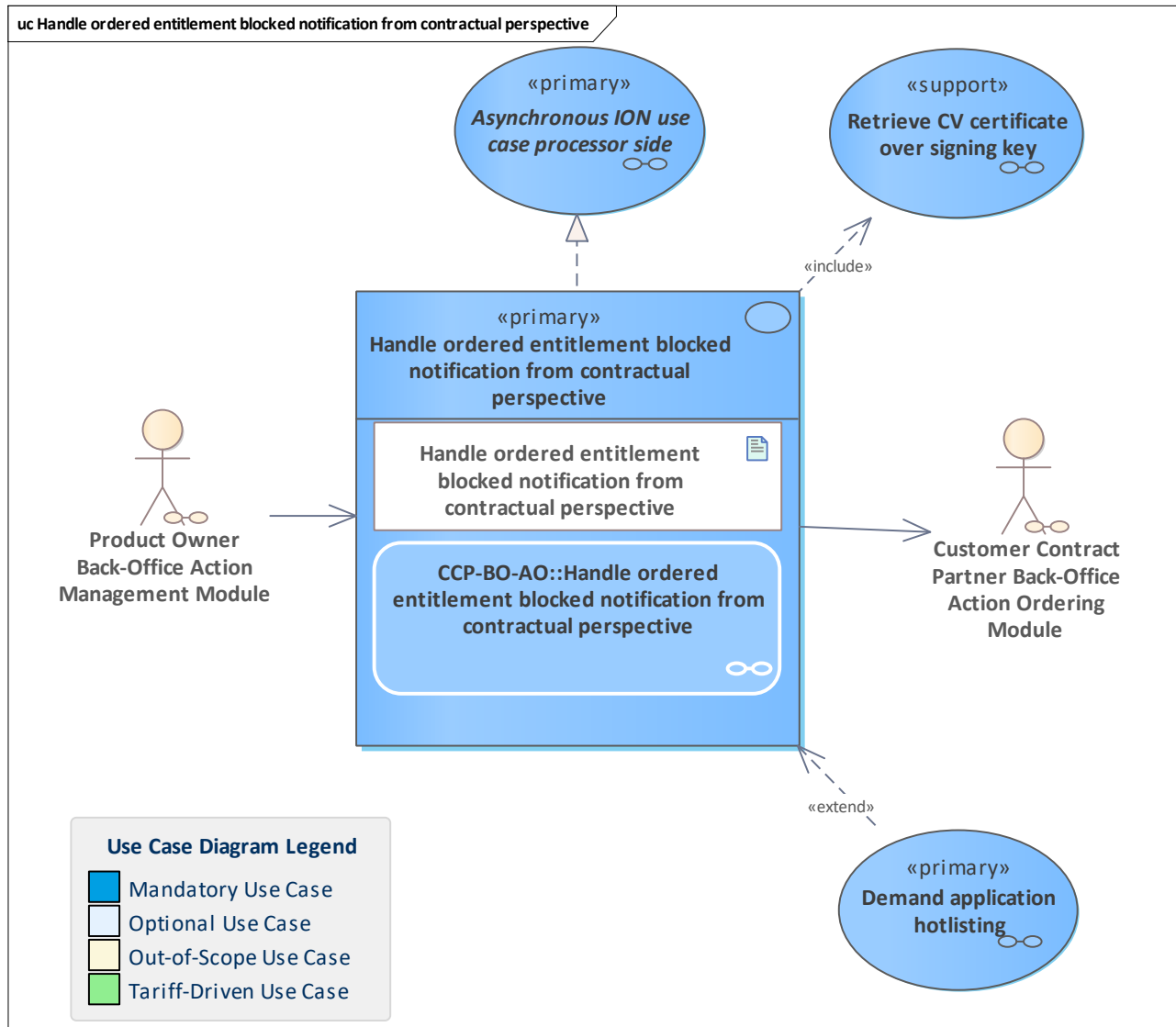


<b>Use Case</b>	<u>Handle ordered entitlement terminated notification from contractual perspective</u>
<b>Description</b>	<p>Handle the notification about a successful execution of an entitlement termination ordered via the action management from the contractual perspective.</p> <p>The ordered entitlement terminated notification is received by the ordering CCP.</p> <p>The CCP does its checks and monitoring from the contractual perspective regarding the correct execution of the termination. In this context, the signature of the termination attestation is verified.</p> <p><b>Note:</b> this closes a potentially related user account concerning the entitlement.</p> <p>If the termination was correct, the pCCP checks if the entitlement has to be removed from the hotlist.</p>
<b>Initiating Actor</b>	<u>Product Owner Back-Office Action Management Module</u>



<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Remove entitlement from hotlist</a> / <a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward ordered entitlement terminated notification : forwardOrderedEntitlementTerminatedNotification</a>
<b>Outputs</b>	<a href="#">Forward ordered entitlement terminated notification response : forwardOrderedEntitlementTerminatedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward ordered entitlement terminated notification exception : forwardOrderedEntitlementTerminatedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Handle ordered entitlement terminated notification from contractual perspective</a>

## 20.2.11 Handle ordered entitlement blocked notification from contractual perspective

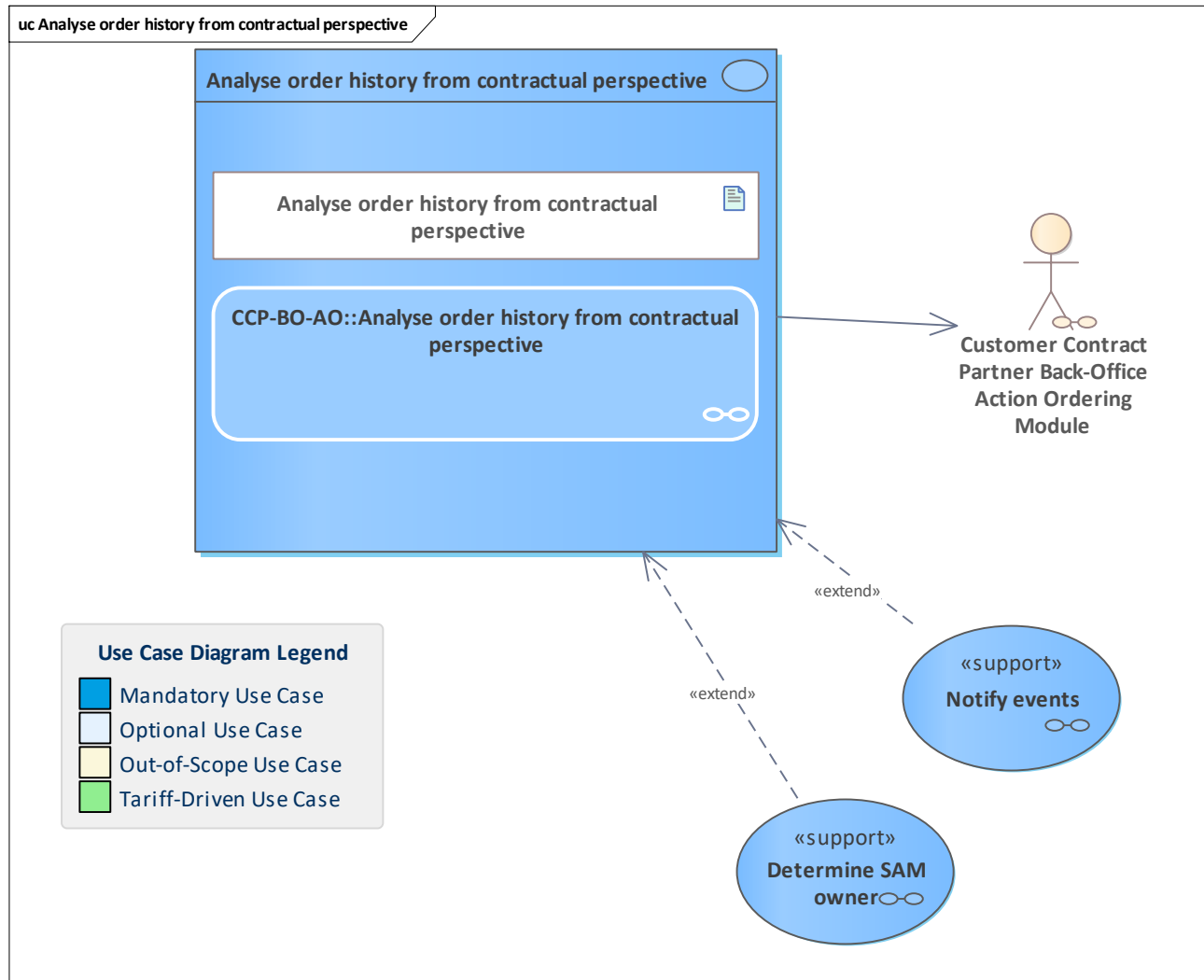


<b>Use Case</b>	<a href="#">Handle ordered entitlement blocked notification from contractual perspective</a>
<b>Description</b>	Handle the notification about a successful execution of an entitlement blocking ordered via the action management from the contractual perspective. The ordered entitlement blocked notification is received by the ordering CCP. The CCP does its checks and monitoring from the contractual perspective regarding the correct execution of the blocking. In this context, the signature of the embedded attestation is verified.
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Action Management Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	



<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward ordered entitlement blocked notification : forwardOrderedEntitlementBlockedNotification</a>
<b>Outputs</b>	<a href="#">Forward ordered entitlement blocked notification response : forwardOrderedEntitlementBlockedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward ordered entitlement blocked notification exception : forwardOrderedEntitlementBlockedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">CCP-BO-AO::Handle ordered entitlement blocked notification from contractual perspective</a>

## 20.2.12 Analyse order history from contractual perspective



<b>Use Case</b>	<a href="#">Analyse order history from contractual perspective</a>
<b>Description</b>	The notifications related to action orders are analysed for correctness.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Action Ordering Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Notify events</a> / <a href="#">Determine SAM owner</a>
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	





<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#"><u>CCP-BO-AO::Analyse order history from contractual perspective</u></a>



## 21 Ordered Action Management Bundle Executing-CCP-System

Functionality bundle that covers use cases to implement CCP back-office system functionality for executing (ordered) actions located in a distributed action list.

### 21.1 Overview

Handle ordered entitlement issued notification from operational perspective

Handle ordered entitlement unblocked notification from operational perspective

Handle ordered entitlement terminated notification from operational perspective

Optional: Handle ordered entitlement blocked notification from operational perspective

Process action list retrieval configuration

Retrieve action list

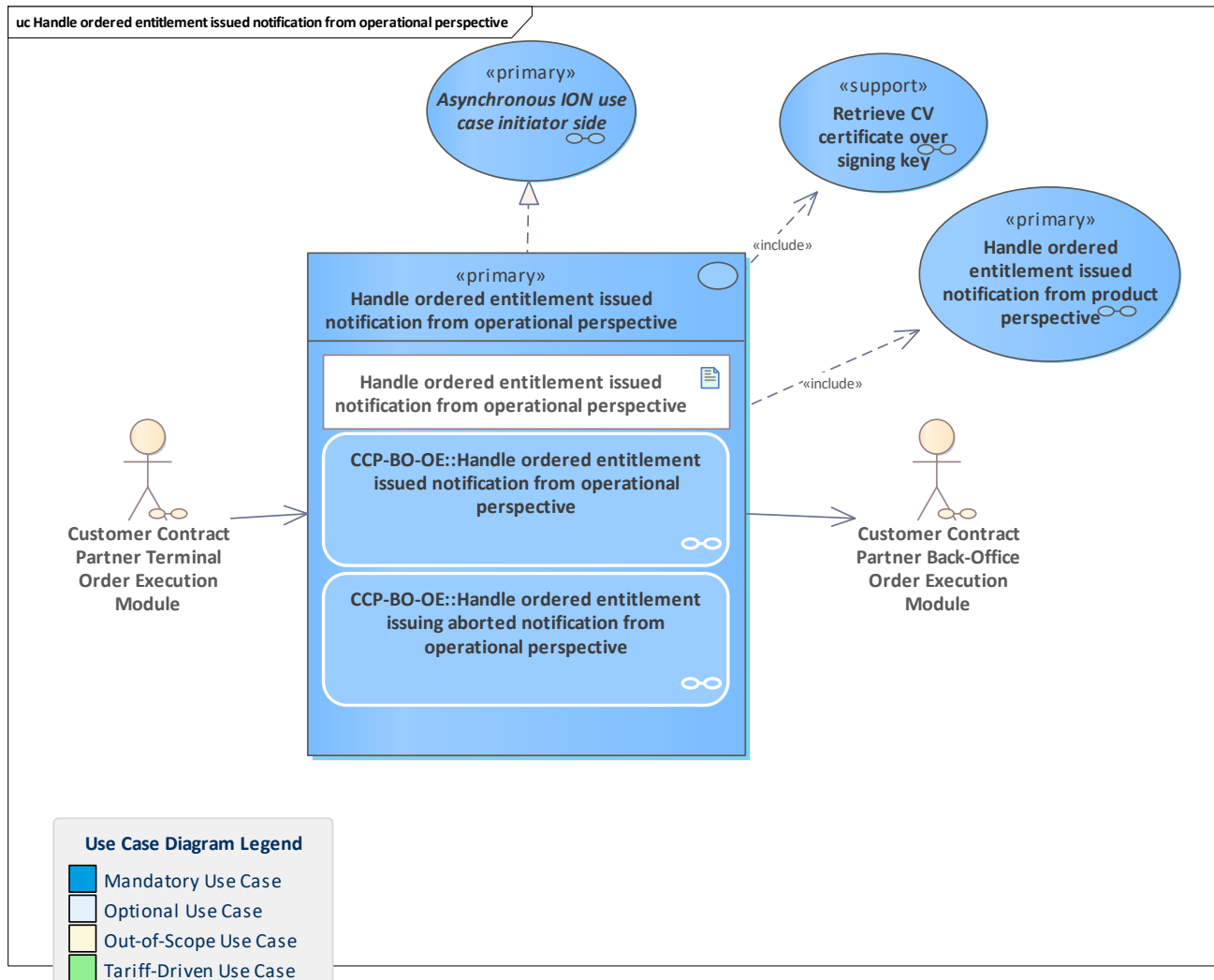
Optional: Retrieve incremental action list

Optional: Verify action list updated via increments

Update action list inventory from operational perspective

### 21.2 Use Cases

#### 21.2.1 Handle ordered entitlement issued notification from operational perspective

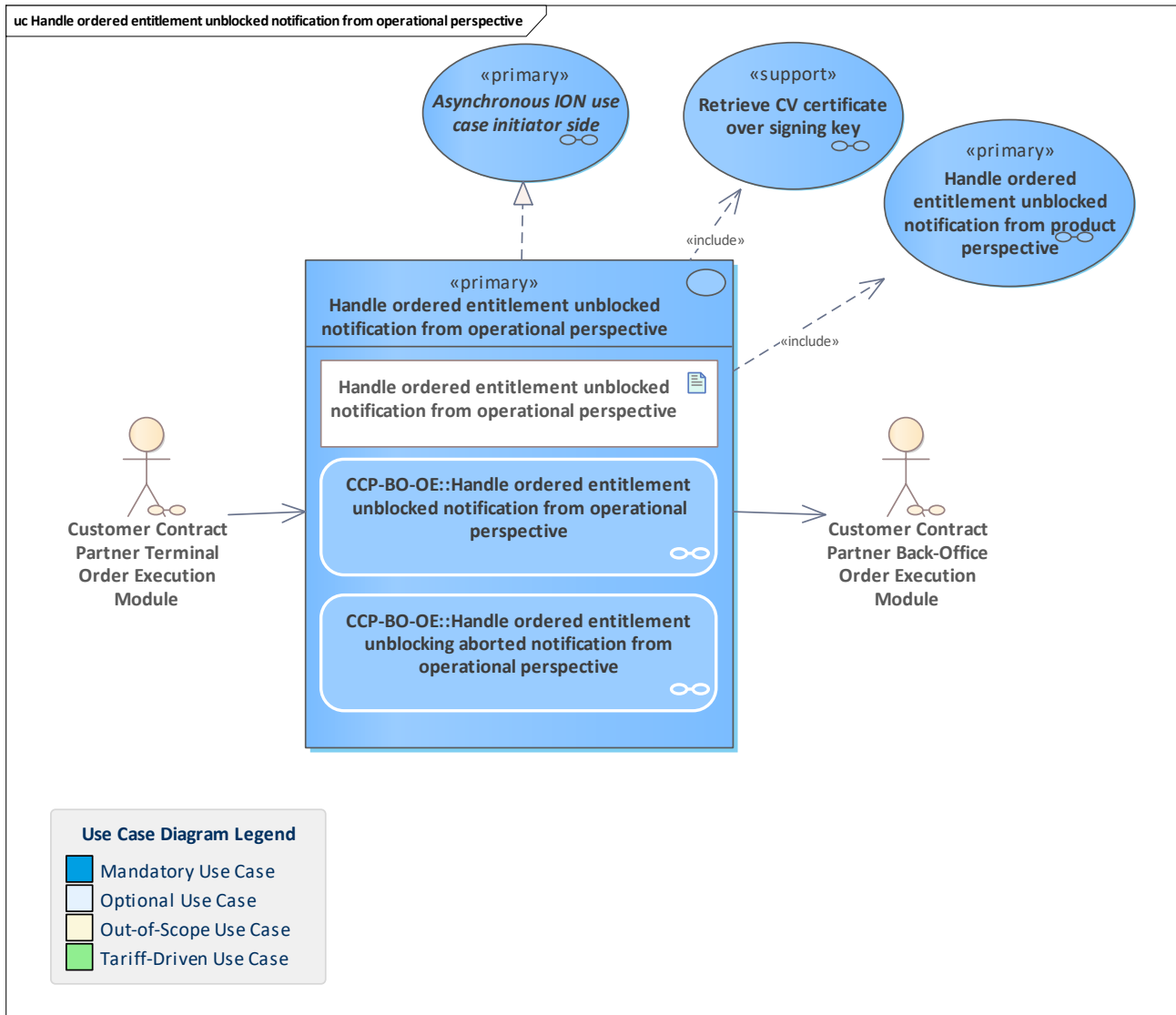


<b>Use Case</b>	<a href="#">Handle ordered entitlement issued notification from operational perspective</a>
<b>Description</b>	<p>Handle the notification about a successful execution of an entitlement issuance ordered via action management from the operational perspective.</p> <p>The executing CCP receives the notification about an entitlement issuance caused by an ordered action and registers it.</p> <p>Then it handles the notification from the operational perspective by performing checks and monitoring, e.g. verifying the signature of the issuance attestation.</p> <p>Finally, the notification is forwarded to the responsible PO system.</p> <p>In the case of action abortion, the notification is also sent to the PO system to announce the used SAM- and product issuance counters.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Order Execution Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Order Execution Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle ordered entitlement issued notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV</a>



	<a href="#">certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify ordered entitlement issued : tNotifyOrderedEntitlementIssued</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Handle ordered entitlement issued notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify ordered entitlement issuing aborted : tNotifyOrderedEntitlementIssuingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Handle ordered entitlement issuing aborted notification from operational perspective</a>

## 21.2.2 Handle ordered entitlement unblocked notification from operational perspective

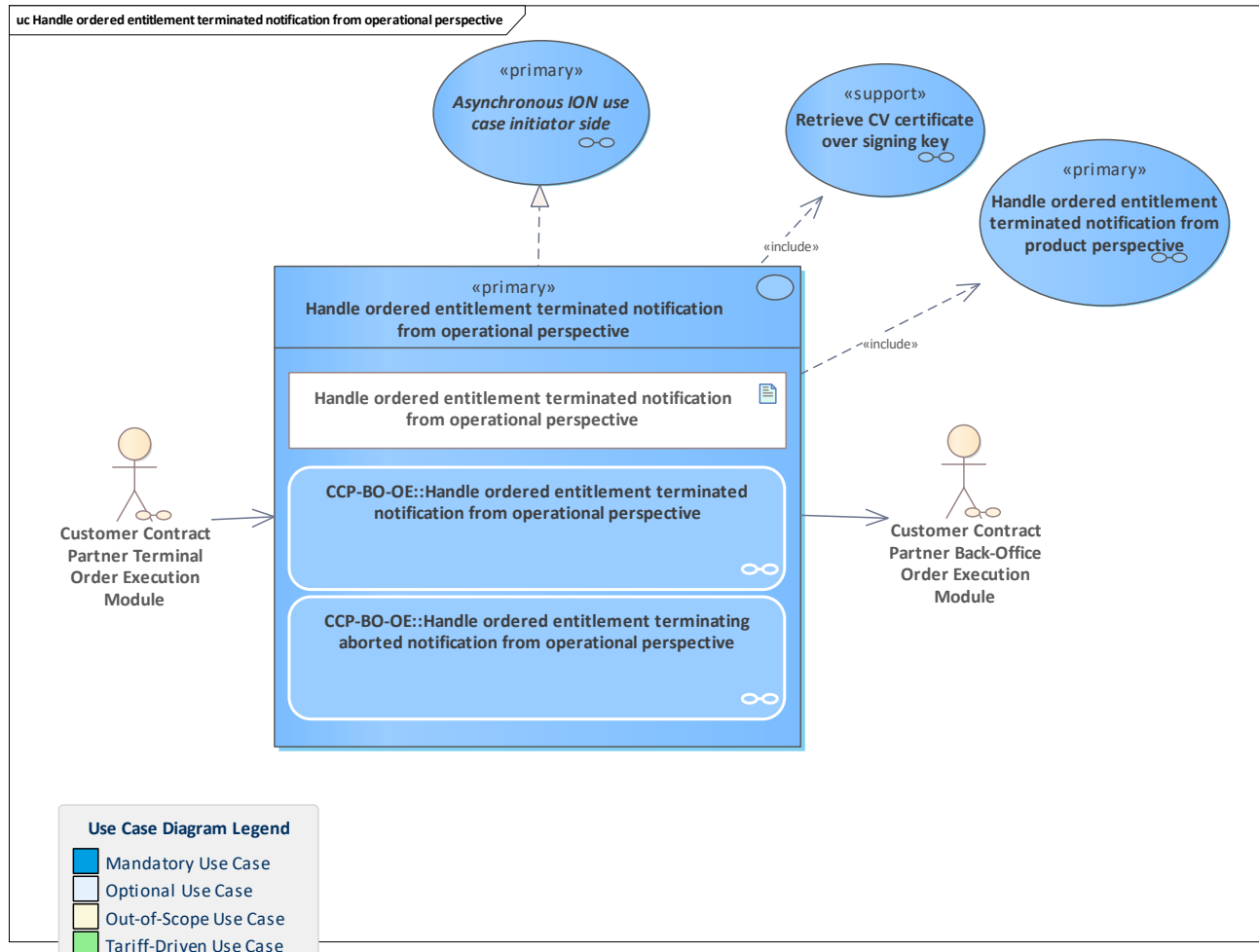


<b>Use Case</b>	<u>Handle ordered entitlement unblocked notification from operational perspective</u>
<b>Description</b>	<p>Handle the notification about a successful execution of an entitlement unblocking ordered via action management from the operational perspective.</p> <p>The ordered entitlement unblocked notification is sent by the CCP terminal to the CCP back-office system. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation.</p> <p>The notification will be sent to the PO (and the PO will forward it later to the ordering CCP).</p> <p>In the case of action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The CCP back-office system registers the abortion for internal monitoring and data consistency.</p>



<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Order Execution Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Order Execution Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle ordered entitlement unblocked notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify ordered entitlement unblocked : tNotifyOrderedEntitlementUnblocked</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Handle ordered entitlement unblocked notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify ordered entitlement unblocking aborted : tNotifyOrderedEntitlementUnblockingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Handle ordered entitlement unblocking aborted notification from operational perspective</a>

## 21.2.3 Handle ordered entitlement terminated notification from operational perspective



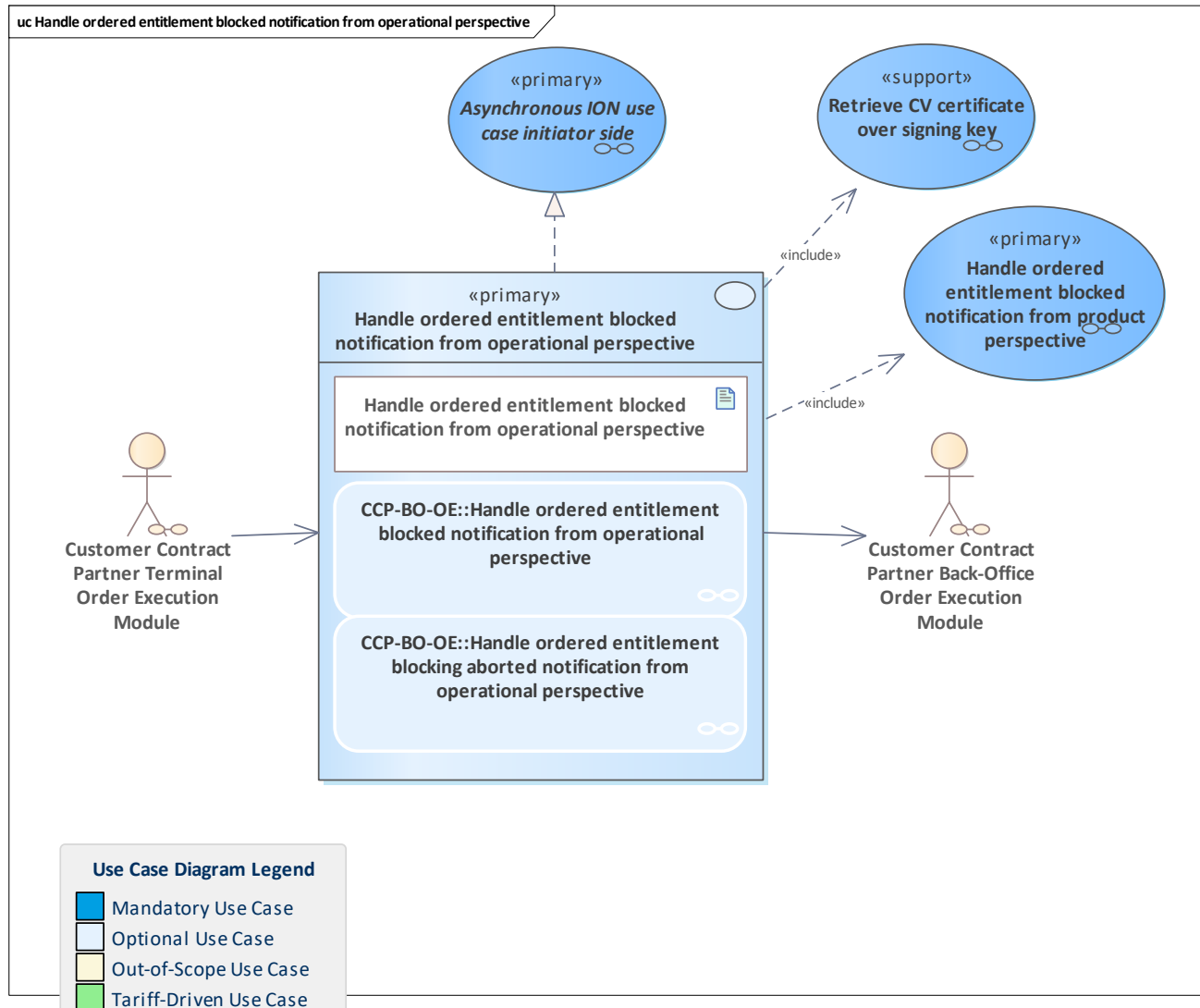
<b>Use Case</b>	<a href="#">Handle ordered entitlement terminated notification from operational perspective</a>
<b>Description</b>	<p>Handle the notification about a successful execution of an entitlement termination ordered via action management from the operational perspective.</p> <p>The entitlement terminated notification is sent by the CCP terminal to the back-office system of the executing CCP. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation.</p> <p>The notification will be sent to the PO (and the PO will forward it later to the ordering CCP).</p> <p>In the case of action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The CCP back-office system registers the abortion for internal monitoring and data consistency.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Order Execution Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Order Execution Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	



<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle ordered entitlement terminated notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify ordered entitlement terminated : tNotifyOrderedEntitlementTerminated</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Handle ordered entitlement terminated notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify ordered entitlement terminating aborted : tNotifyOrderedEntitlementTerminatingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Handle ordered entitlement terminating aborted notification from operational perspective</a>



## 21.2.4 Optional: Handle ordered entitlement blocked notification from operational perspective

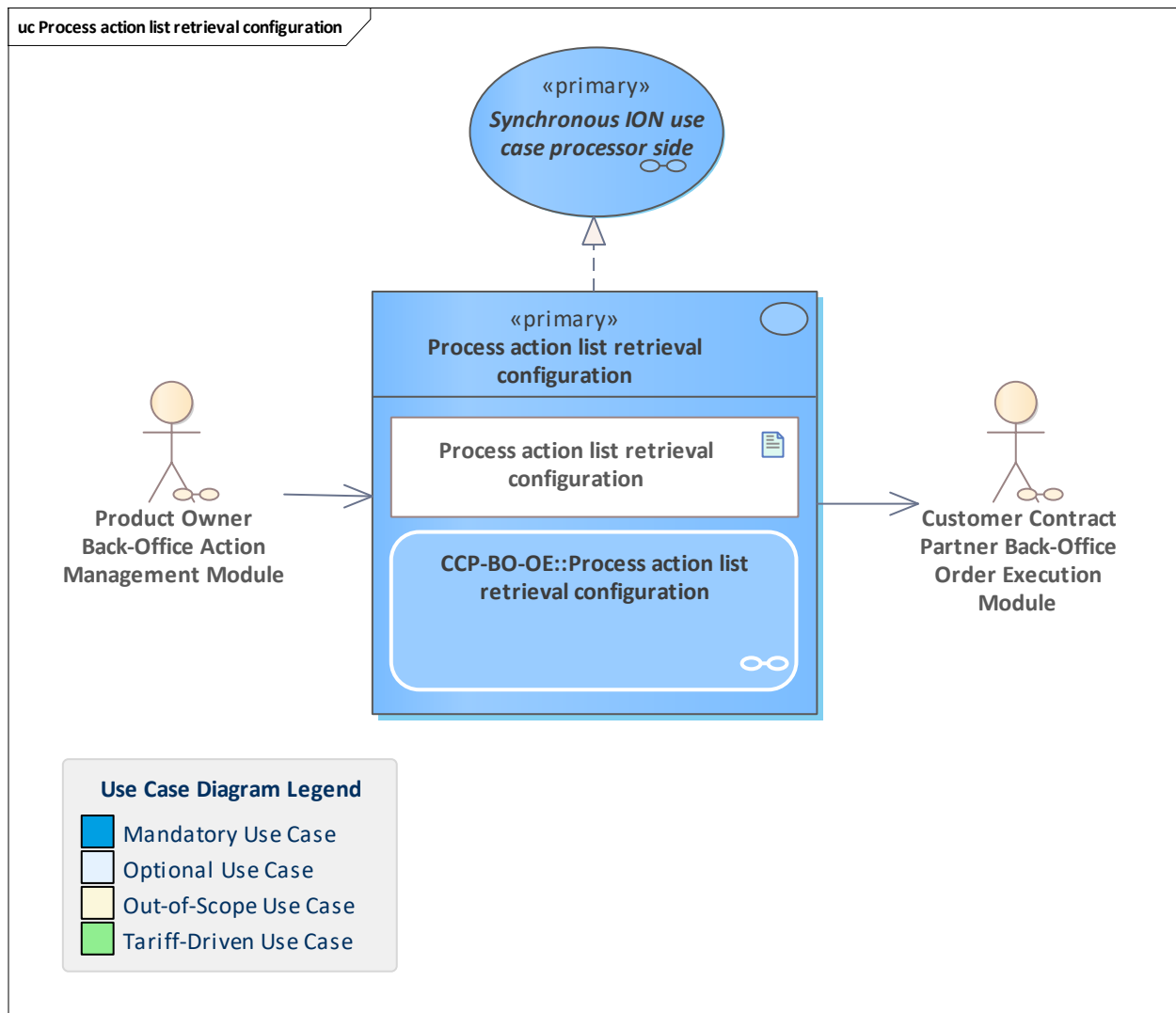


<b>Use Case</b>	<a href="#">Handle ordered entitlement blocked notification from operational perspective</a>
<b>Description</b>	<p>Handle the notification about a successful execution of an entitlement blocking ordered via the action management from the operational perspective.</p> <p>The <i>ordered entitlement blocked notification</i> is sent by the CCP terminal to the CCP back-office system. The notification will be checked and monitored, e.g. by verifying the signature of the embedded attestation.</p> <p>The notification will be sent to the PO (and the PO will forward it later to the ordering CCP)</p> <p>In the case action abortion, terminal and SAM action data is sent by the terminal to the back-office system. The CCP back-office system registers the abortion for internal monitoring and data consistency.</p>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Order Execution Module</a>



<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Order Execution Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle ordered entitlement blocked notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a> / <a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify ordered entitlement blocked : tNotifyOrderedEntitlementBlocked</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Handle ordered entitlement blocked notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify ordered entitlement blocking aborted : tNotifyOrderedEntitlementBlockingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Handle ordered entitlement blocking aborted notification from operational perspective</a>

## 21.2.5 Process action list retrieval configuration

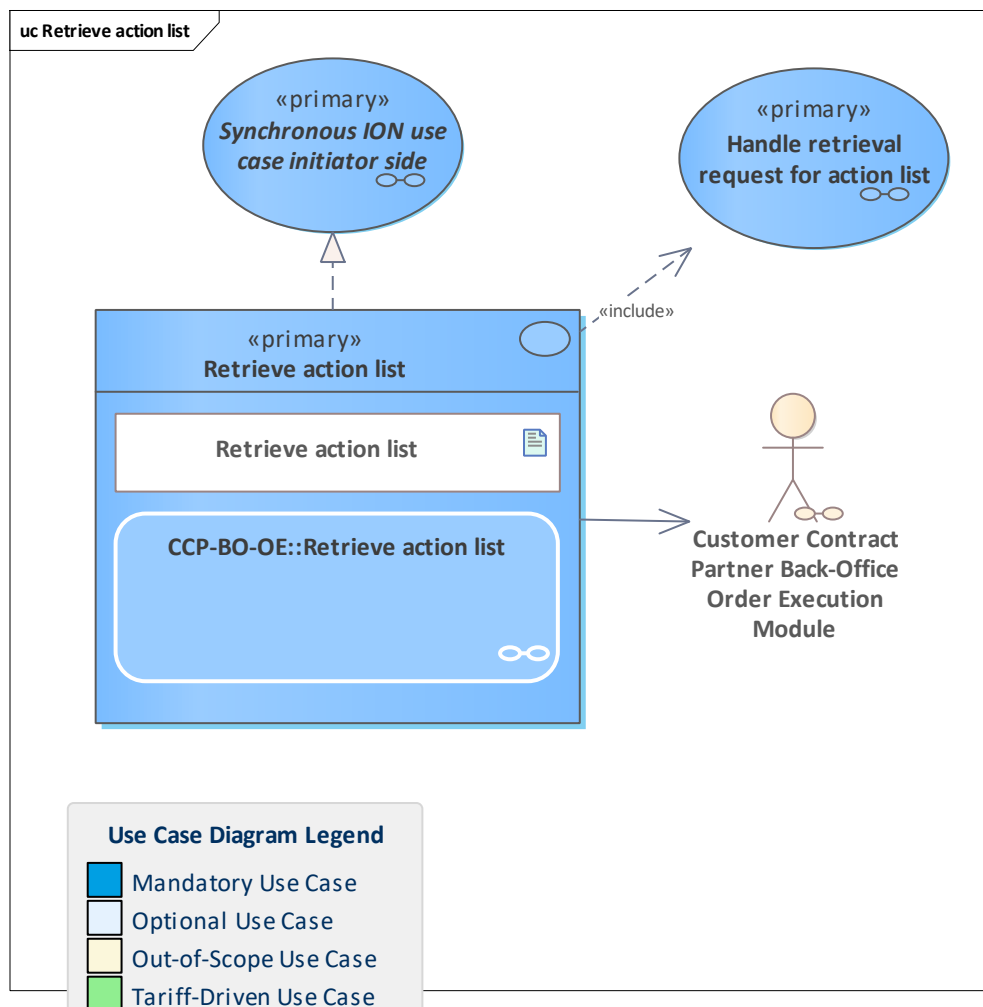


<b>Use Case</b>	<a href="#">Process action list retrieval configuration</a>
<b>Description</b>	The action list retrieval configuration provided by the <a href="#">Product Owner Back-Office Action Management Module</a> is stored in the <a href="#">Customer Contract Partner Back-Office Order Execution Module</a> . The configuration contains information about whether and when the action list is provided in an updated form. It overwrites any previous configurations.
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Action Management Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Order Execution Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case processor side</a>



Base Activity	
Inputs	<a href="#">Put action list retrieval configuration :</a> <a href="#">putActionListRetrievalConfiguration</a>
Outputs	<a href="#">Put action list retrieval configuration response :</a> <a href="#">putActionListRetrievalConfigurationResponse</a>
Error Cases	<a href="#">Put action list retrieval configuration exception :</a> <a href="#">putActionListRetrievalConfigurationException</a>
Activity Diagram	<a href="#">CCP-BO-OE::Process action list retrieval configuration</a>

## 21.2.6 Retrieve action list

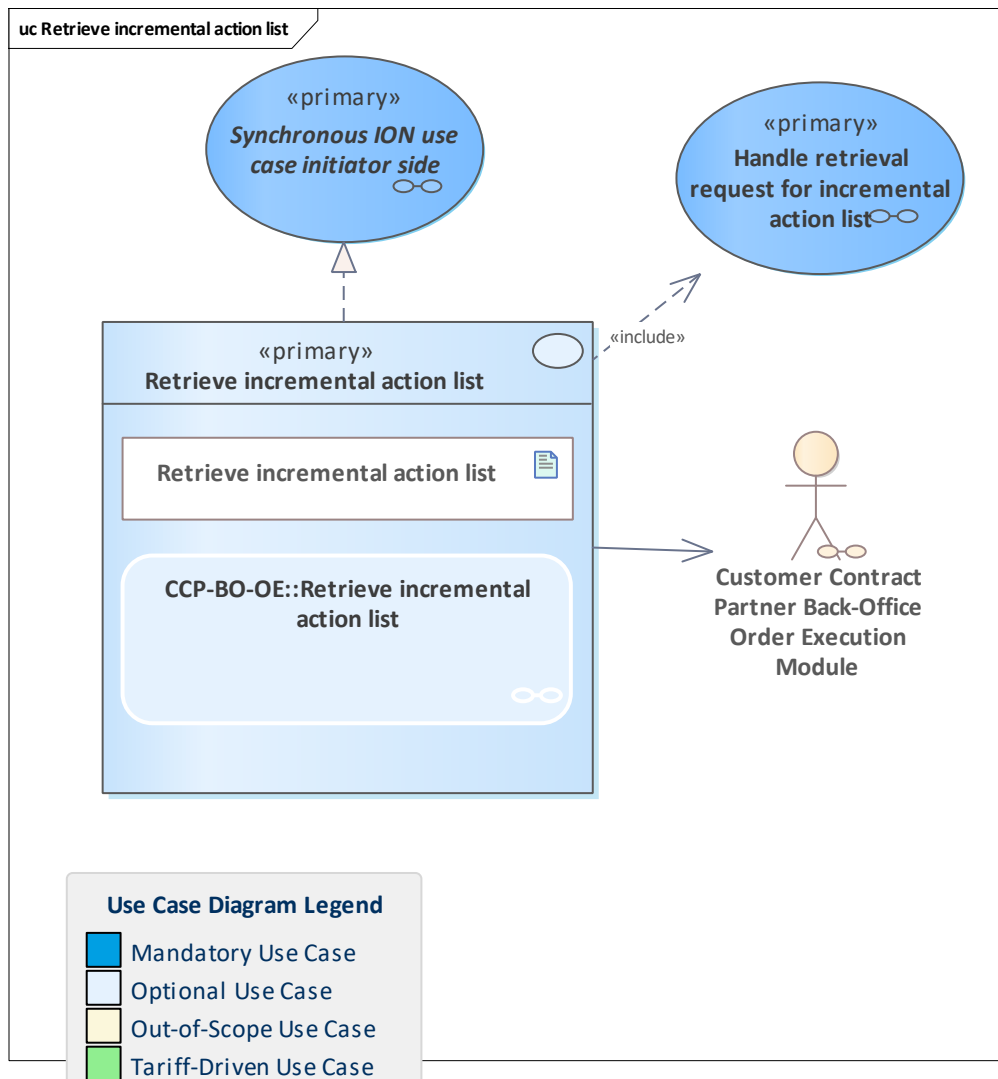


<b>Use Case</b>	<a href="#">Retrieve action list</a>
<b>Description</b>	The <a href="#">Customer Contract Partner Back-Office Order Execution Module</a> retrieves the latest action list from a <a href="#">Product Owner Back-Office Action Management Module</a> .
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Order Execution Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for action list</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Action list : actionList</a> <a href="#">Process instance ID : ProcessInstanceId</a> <a href="#">Organisation ID of action list providing system : OrganisationId</a>
<b>Outputs</b>	



<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#"><u>CCP-BO-OE::Retrieve action list</u></a>

## 21.2.7 Optional: Retrieve incremental action list



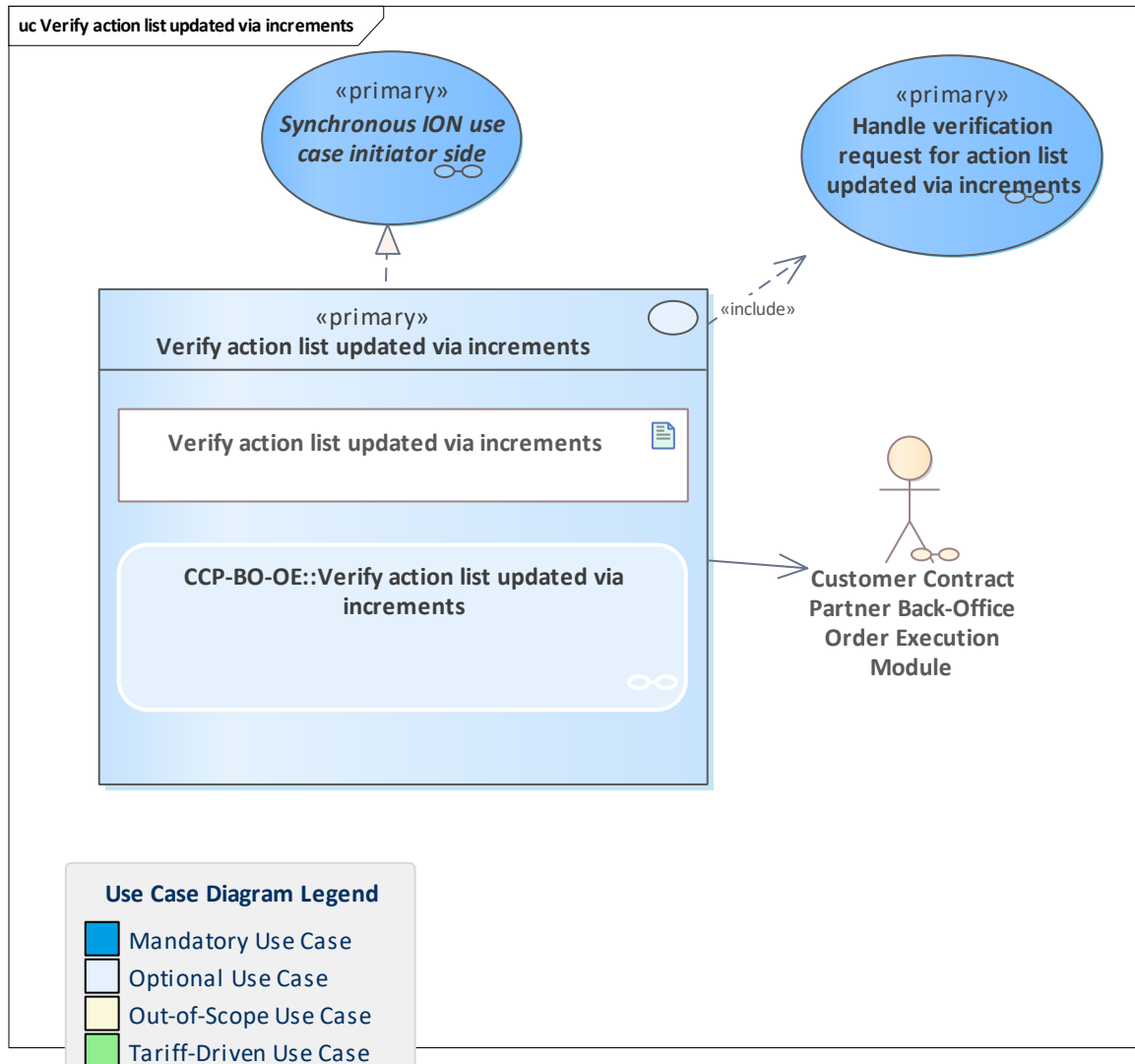
<b>Use Case</b>	<a href="#">Retrieve incremental action list</a>
<b>Description</b>	The <a href="#">Customer Contract Partner Back-Office Order Execution Module</a> retrieves the incremental action list including the increments since the cycle in its inventory from a <a href="#">Product Owner Back-Office Action Management Module</a> .
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Order Execution Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle retrieval request for incremental action list</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Organisation ID of action list providing system : OrganisationId</a> <a href="#">Process instance ID : ProcessInstanceId</a>



	<a href="#">Updated action list : actionList</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Retrieve incremental action list</a>



## 21.2.8 Optional: Verify action list updated via increments

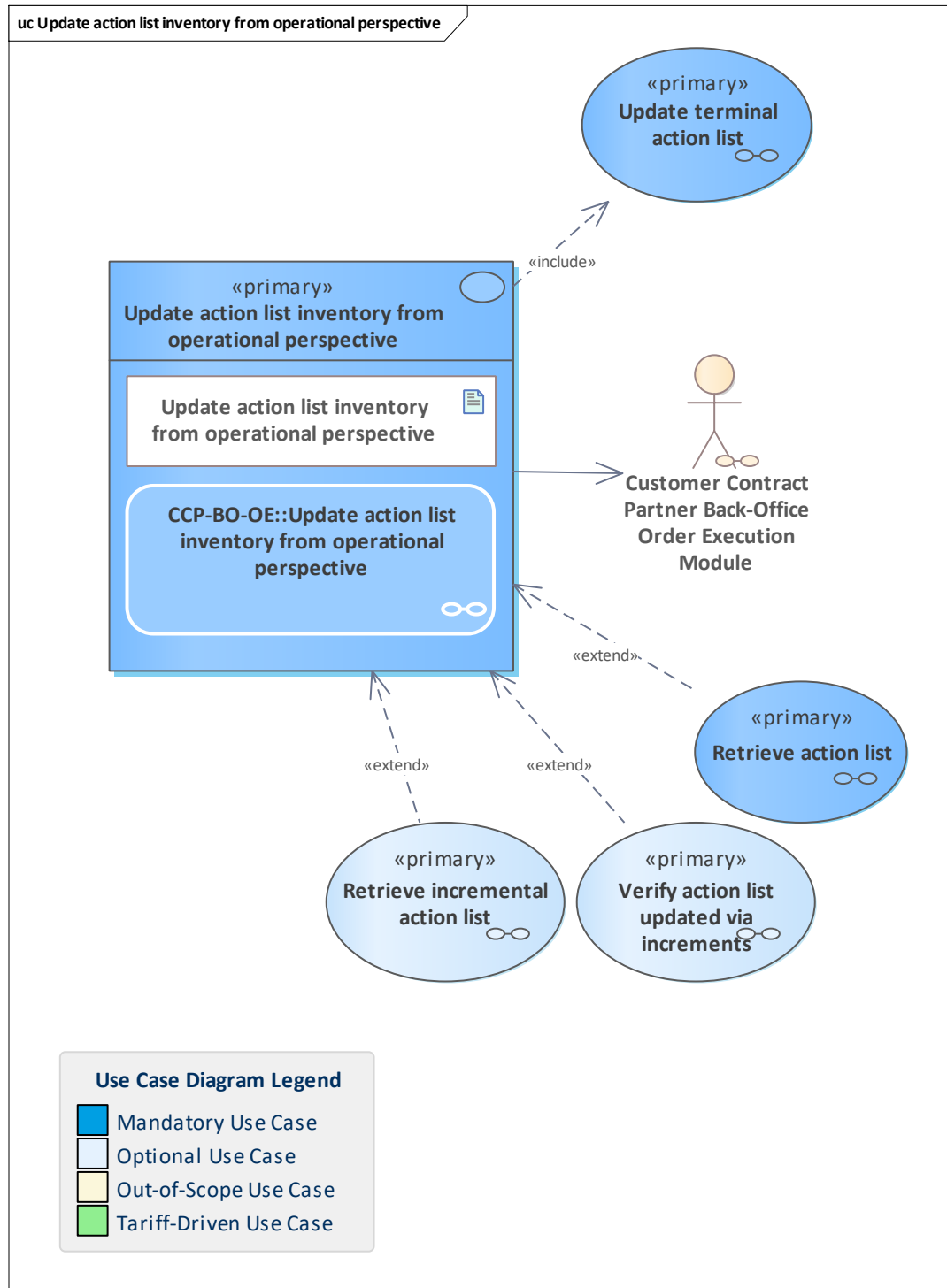


<b>Use Case</b>	<a href="#">Verify action list updated via increments</a>
<b>Description</b>	The <a href="#">Customer Contract Partner Back-Office Order Execution Module</a> verifies that its inventory is consistent with the full list by verifying the checksum after the inventory is updated via increments. See also <a href="#">Checksum calculation for hotlist and action list verification</a> and <a href="#">Example calculation for an action list inventory</a> .
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Order Execution Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle verification request for action list updated via increments</a>
<b>Linked Use Cases</b>	<a href="#">Synchronous ION use case initiator side</a>



<b>(Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Process instance ID : ProcessInstanceId</a> <a href="#">Updated action list : actionList</a> <a href="#">Organisation ID of action list providing system : OrganisationId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Verify action list updated via increments</a>

## 21.2.9 Update action list inventory from operational perspective



<b>Use Case</b>	<u>Update action list inventory from operational perspective</u>
<b>Description</b>	The action list is retrieved by the <u>Customer Contract Partner Back-Office Order Execution Module</u> from the <u>Product Owner Back-Office Action Management Module</u> and distributed to its relevant



	<p>terminals.</p> <p>When determining when to invoke this process, the retrieval configuration provided by the <a href="#">Product Owner Back-Office Action Management Module</a> should be respected, see <a href="#">Process action list retrieval configuration</a>.</p> <p>When several <a href="#">Product Owner Back-Office Action Management Modules</a> are involved, the updated action lists need to be merged before passing them on to the terminals. This scenario is not shown here for simplicity.</p>
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Order Execution Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Verify action list updated via increments</a> / <a href="#">Retrieve incremental action list</a> / <a href="#">Retrieve action list</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Update terminal action list</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Organisation ID of action list providing system : OrganisationId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-OE::Update action list inventory from operational perspective</a>



## 22 Static Entitlements Bundle CCP-System

Functionality bundle that covers CCP back-office system use cases for working with static entitlements.

### 22.1 Overview

Handle static entitlement issued notification from operational perspective

Handle static entitlement issued notification from contractual perspective

Handle static entitlement inspected notification from contractual perspective

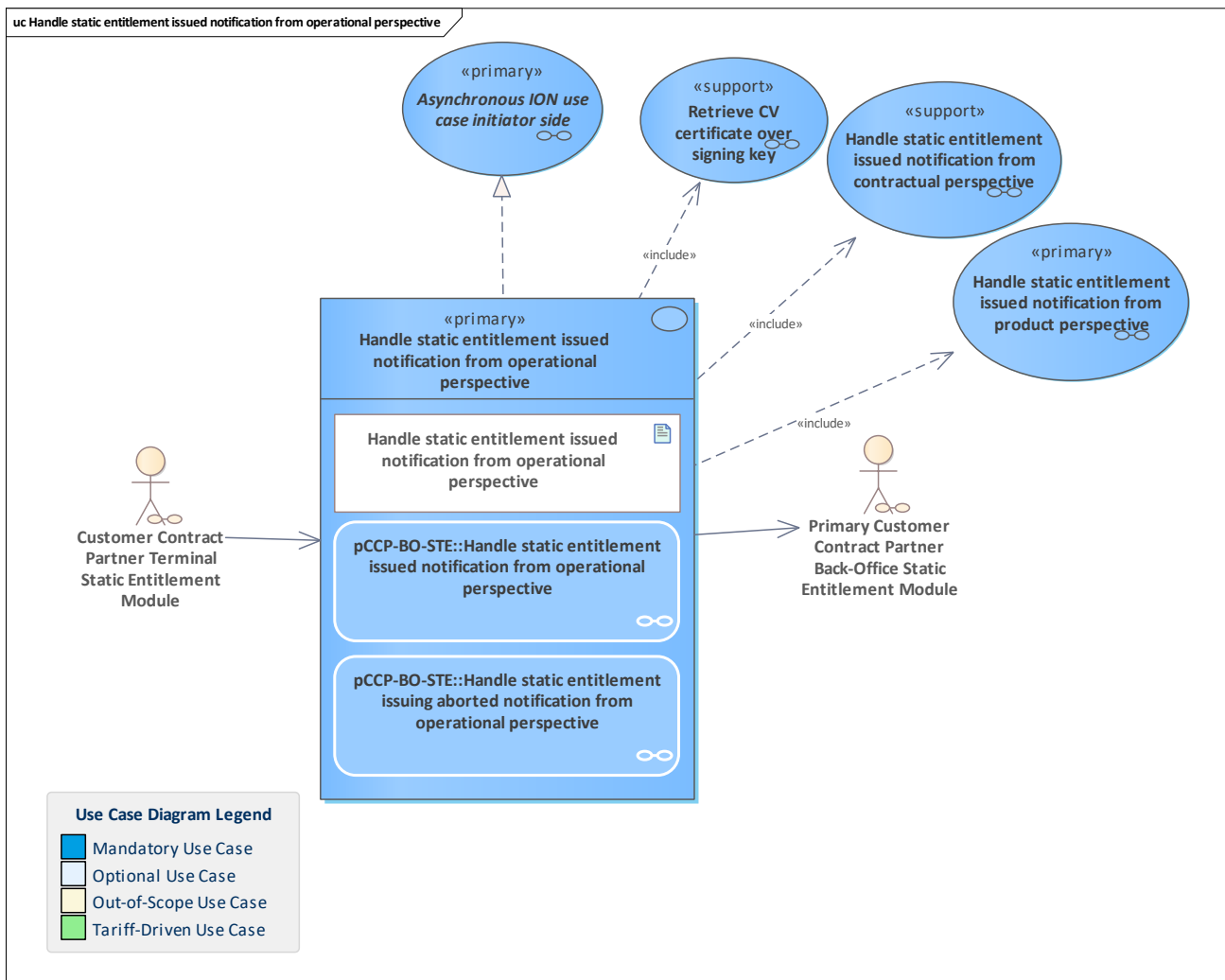
Handle static entitlement terminated notification from operational perspective

Handle static entitlement terminated notification from contractual perspective

Check static entitlement notifications against issuance notification from contractual perspective

### 22.2 Use Cases

#### 22.2.1 Handle static entitlement issued notification from operational perspective

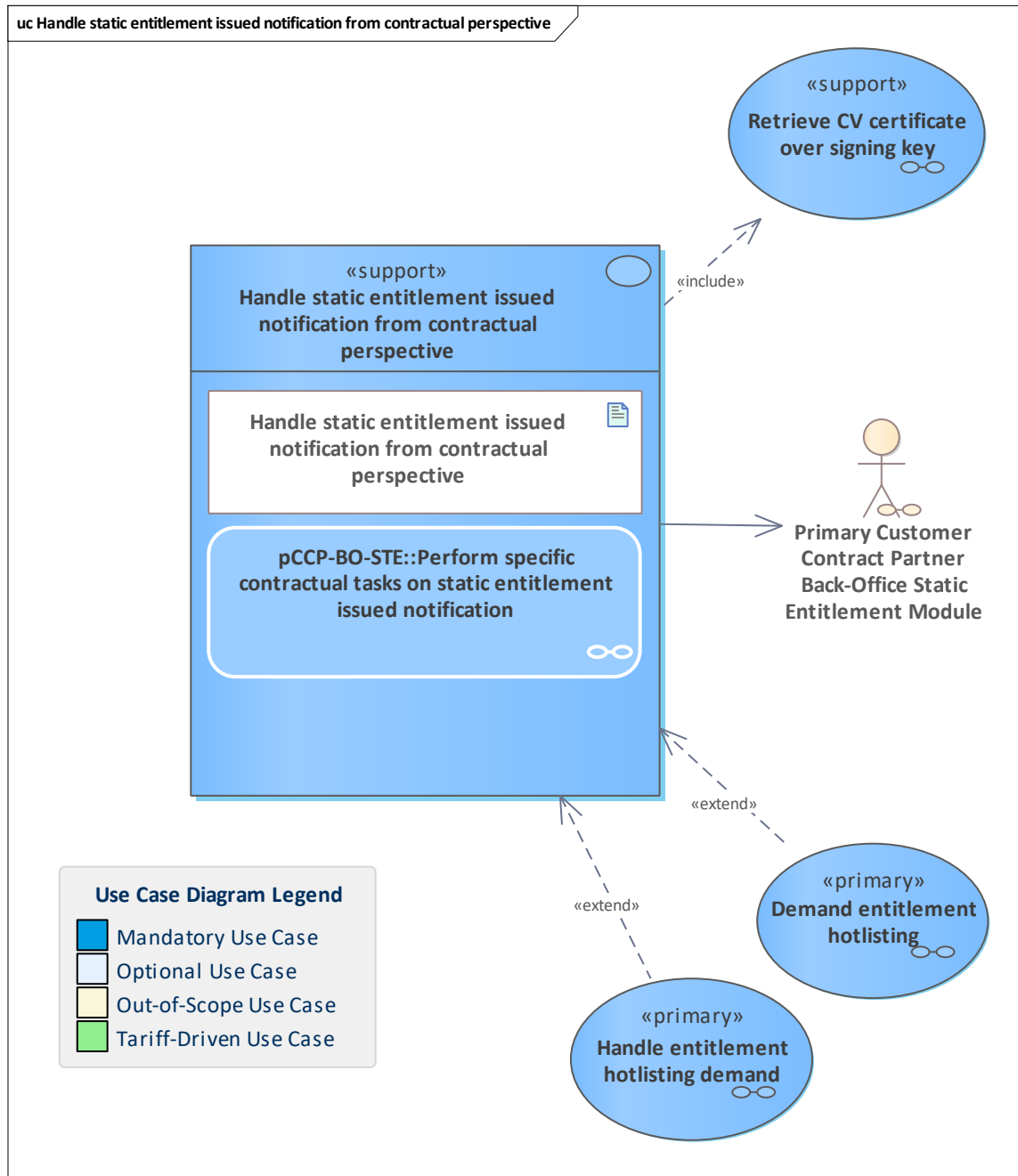


<b>Use Case</b>	<u>Handle static entitlement issued notification from operational perspective</u>
<b>Description</b>	<p>Handle a notification about a static entitlement issuance from the operational perspective.</p> <p>The static entitlement extension of the CCP terminal sends the notification about the issuance of a static entitlement to the responsible pCCP back-office system with static entitlement extension.</p> <p>The pCCP system does its operational checks and monitoring, such as the SAM signature verification of the static entitlement. Then, the notification is sent to the responsible PO system with a static entitlement extension. This can be done either via a single message or in a scheduled process as a message list. Since it is the pCCP, the system does the contractual checks and monitoring, too.</p> <p>This use case also handles a possible abortion of the entitlement issuance. Also in this case, the registration and forwarding of the notification to the PO system is important due to the SAM and product issuance counters, to keep the monitoring consistent.</p>
<b>Initiating Actor</b>	<u>Customer Contract Partner Terminal Static Entitlement Module</u>
<b>Reacting Actor</b>	<u>Primary Customer Contract Partner Back-Office Static Entitlement Module</u>



<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle static entitlement issued notification from product perspective</a> / <a href="#">Handle static entitlement issued notification from contractual perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Terminal notify static entitlement issued : tNotifyStaticEntitlementIssued</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-STE::Handle static entitlement issued notification from operational perspective</a>
<b>Alternative 1</b>	
<b>Inputs</b>	<a href="#">Terminal notify static entitlement issued aborted : tNotifyStaticEntitlementIssuingAborted</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-STE::Handle static entitlement issuing aborted notification from operational perspective</a>

## 22.2.2 Handle static entitlement issued notification from contractual perspective



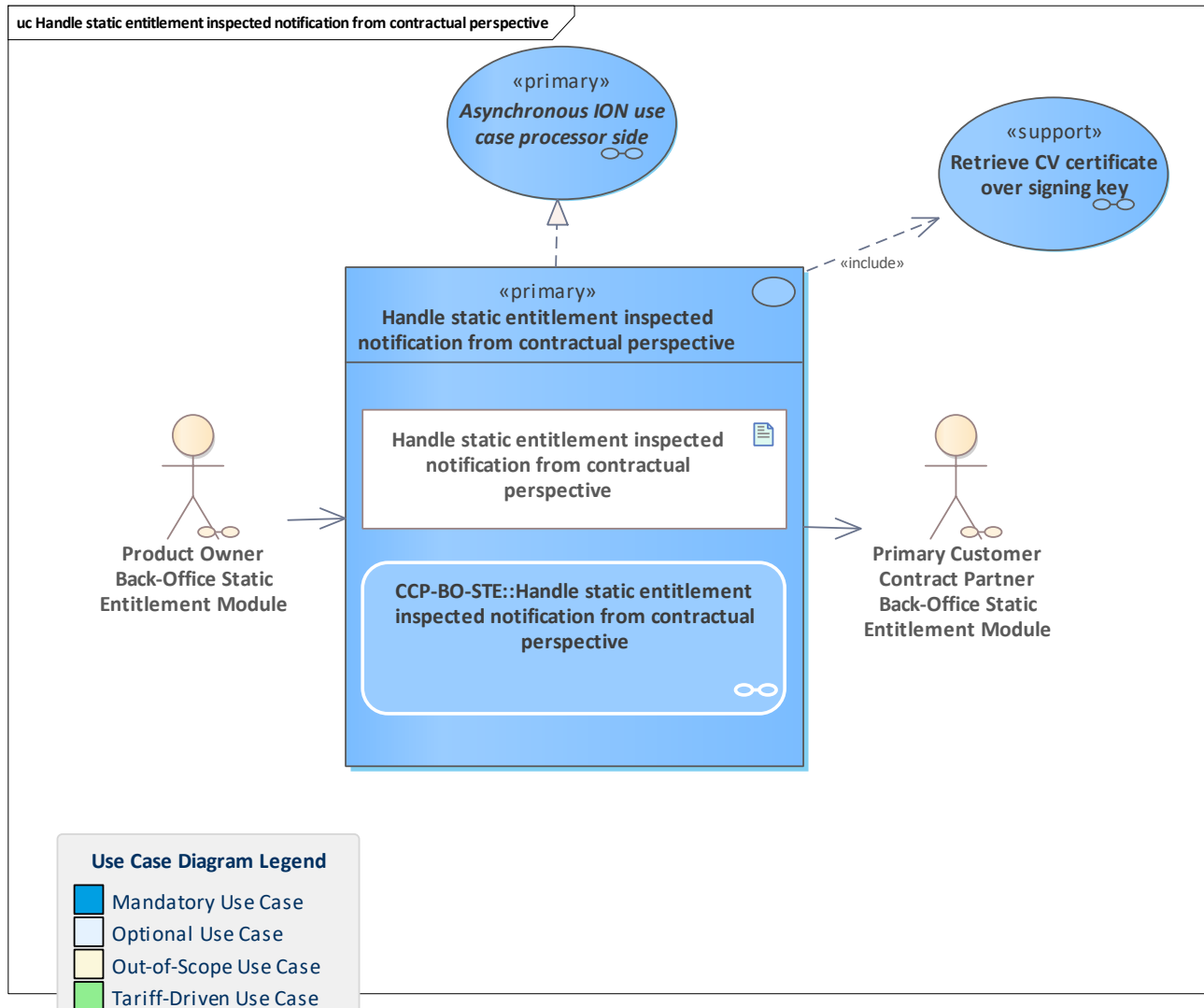
<b>Use Case</b>	<a href="#">Handle static entitlement issued notification from contractual perspective</a>
<b>Description</b>	Handle a notification about an issuance of an owned static entitlement from the contractual perspective. All needed checks and monitoring are performed.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Static Entitlement Module</a>





<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand entitlement hotlisting</a> / <a href="#">Handle entitlement hotlisting demand</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement hotlisting demand</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-STE::Perform specific contractual tasks on static entitlement issued notification</a>

## 22.2.3 Handle static entitlement inspected notification from contractual perspective

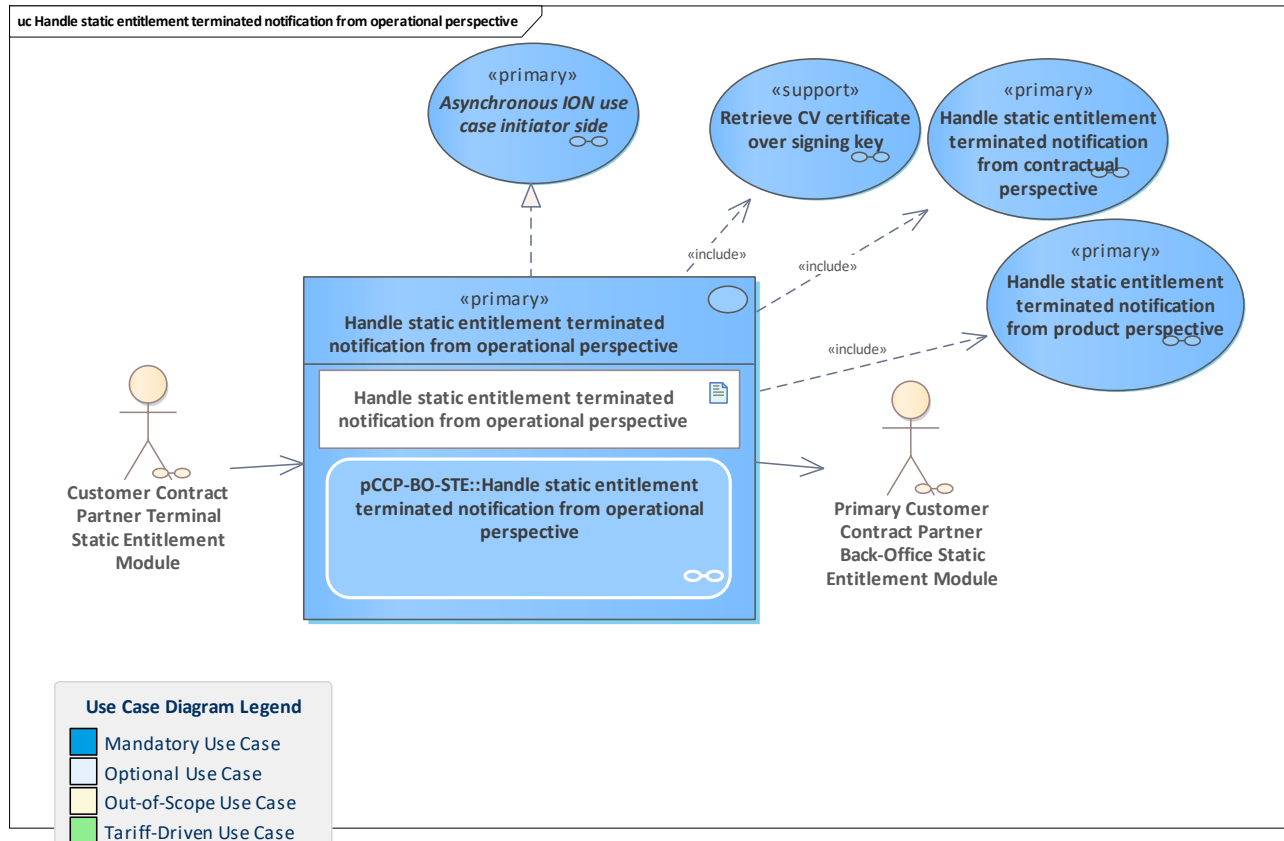


<b>Use Case</b>	<a href="#">Handle static entitlement inspected notification from contractual perspective</a>
<b>Description</b>	This use case describes the processing of the notification of an inspected static entitlement in the pCCP back-office system. The pCCP receives the notification from the PO system, registers it and does its contractual monitoring checks.
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Static Entitlement Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Static Entitlement Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases</b>	<a href="#">Asynchronous ION use case processor side</a>



<b>(Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<u>Forward static entitlement inspected notification :</u> <u>forwardStaticEntitlementInspectedNotification</u>
<b>Outputs</b>	<u>Forward static entitlements inspected notification response :</u> <u>forwardStaticEntitlementInspectedNotificationResponse</u>
<b>Error Cases</b>	<u>E CCP RECEIVER IS NOT ENTITY OWNER</u> <u>E CO BINARY STRUCTURE CANNOT BE PROCESSED</u> <u>E STE DUPLICATE STATIC ENTITLEMENT ISSUANCE</u> <u>E STE DUPLICATE STATIC ENTITLEMENT TERMINATION</u> <u>E CO WRONG SECURITY LEVEL</u> <u>Forward static entitlements inspected notification exception :</u> <u>forwardStaticEntitlementInspectedNotificationException</u>
<b>Activity Diagram</b>	<u>CCP-BO-STE::Handle static entitlement inspected notification from contractual perspective</u>

## 22.2.4 Handle static entitlement terminated notification from operational perspective

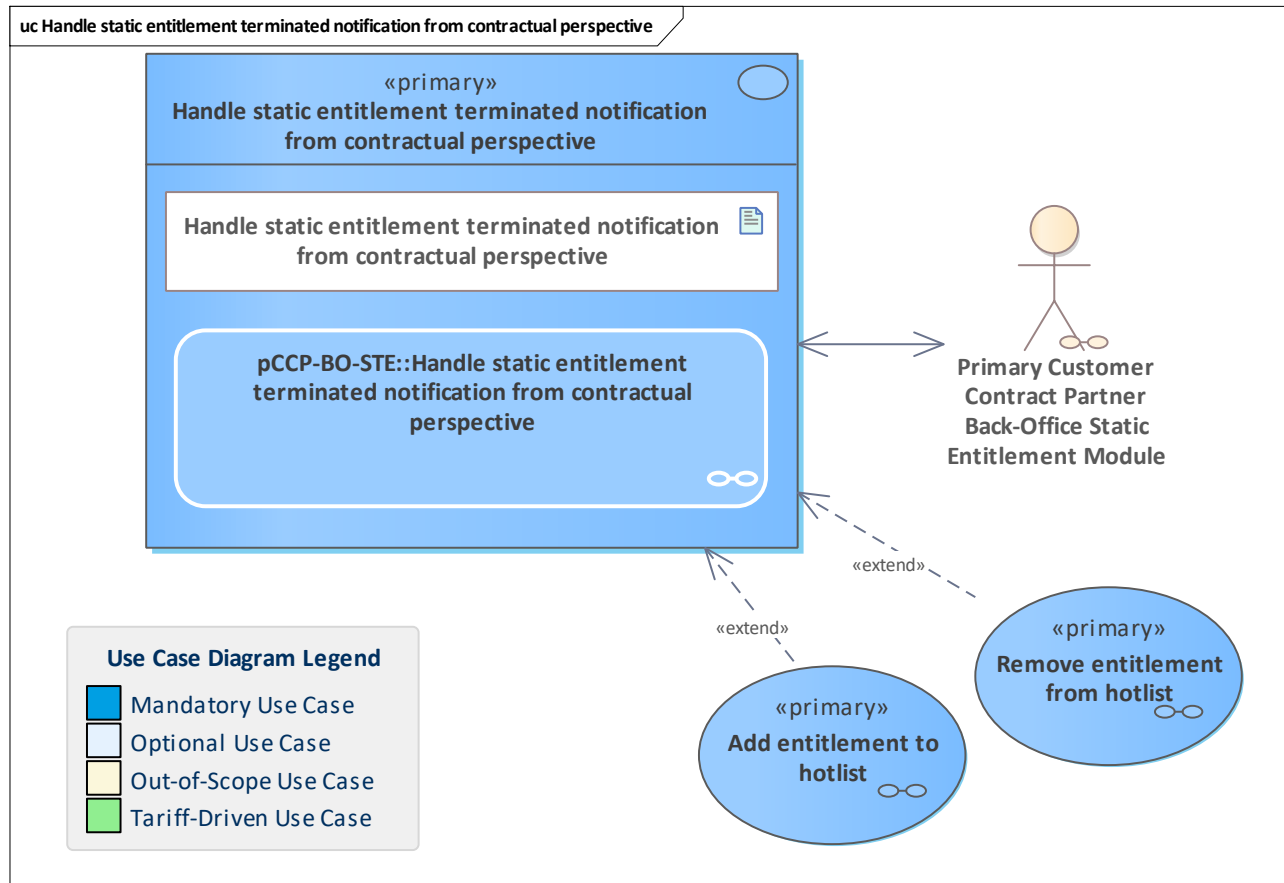


<b>Use Case</b>	<a href="#">Handle static entitlement terminated notification from operational perspective</a>
<b>Description</b>	<p>Handle a static entitlement terminated notification from the operational perspective.</p> <p>The entitlement terminated notification is sent by the CCP terminal to the back-office system of the CCP with a static entitlement extension. The notification will be checked and monitored.</p> <p>Only the pCCP can terminate its own static entitlement:</p> <ul style="list-style-type: none"> <li>the notification will be sent to the PO</li> <li>the pCCP does its contractual checks and monitoring</li> </ul>
<b>Initiating Actor</b>	<a href="#">Customer Contract Partner Terminal Static Entitlement Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Static Entitlement Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle static entitlement terminated notification from contractual perspective</a> / <a href="#">Handle static entitlement terminated notification from product perspective</a> / <a href="#">Retrieve CV certificate over signing key</a> / <a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case initiator side</a>



<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Notify static entitlement terminated from terminal :</a> <a href="#">tNotifyStaticEntitlementTerminated</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-STE::Handle static entitlement terminated notification from operational perspective</a>

## 22.2.5 Handle static entitlement terminated notification from contractual perspective

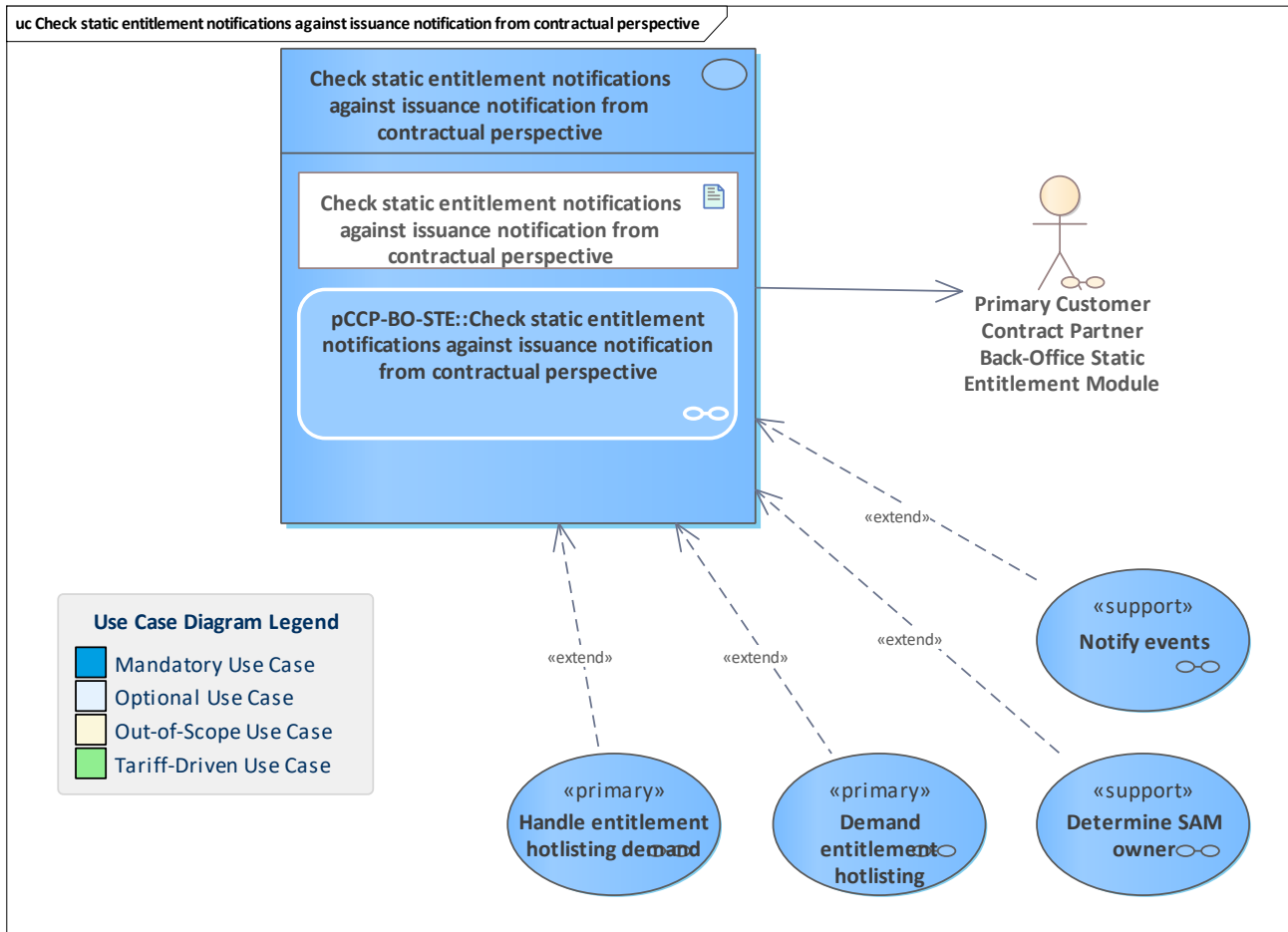


<b>Use Case</b>	<u>Handle static entitlement terminated notification from contractual perspective</u>
<b>Description</b>	<p>The notification of a terminated static entitlement is handled from the contractual perspective.</p> <p>The pCCP back-office system with static entitlement extension does its contractual checks and monitoring.</p> <p>Note: this closes a potentially related user account concerning the static entitlement.</p> <p>The static entitlement must be hotlisted (if it was not already hotlisted) to avoid illegal usage of the static entitlement.</p> <p>Note: For a temporarily hotlisted static entitlement, the hotlist entry must be changed to a non-temporarily hotlist entry by removing it and adding it again with a new hotlist expiry date to the hotlist.</p> <p>Note: the contractual handling of the static entitlement termination is done by the <a href="#">Primary Customer Contract Partner Back-Office Static Entitlement Module</a>; the hotlist activities are done by the <a href="#">Primary Customer Contract Partner Back-Office Main Module</a>.</p>
<b>Initiating Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Static Entitlement Module</a>
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Static Entitlement Module</a>



<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Remove entitlement from hotlist</a> / <a href="#">Add entitlement to hotlist</a>
<b>Linked Use Cases (Includes)</b>	
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Static entitlement notification : StaticEntitlementTerminatedNotification</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-STE::Handle static entitlement terminated notification from contractual perspective</a>

## 22.2.6 Check static entitlement notifications against issuance notification from contractual perspective



<b>Use Case</b>	<a href="#">Check static entitlement notifications against issuance notification from contractual perspective</a>
<b>Description</b>	Non-issuing static entitlement notifications need to be checked for consistency with their issuance notification, e.g., the entitlement validity period. This also makes sure that every entitlement seen live is known to the system or has been reported as issued.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Primary Customer Contract Partner Back-Office Static Entitlement Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Notify events</a> / <a href="#">Demand entitlement hotlisting</a> / <a href="#">Determine SAM owner</a> / <a href="#">Handle entitlement hotlisting demand</a>
<b>Linked Use Cases (Includes)</b>	<a href="#">Handle entitlement hotlisting demand</a>
<b>Linked Use Cases (Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	





<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">pCCP-BO-STE::Check static entitlement notifications against issuance notification from contractual perspective</a>



## 23 Miscellaneous Bundle CCP-System

Functionality bundle that covers miscellaneous CCP back-office system use cases.

### 23.1 Overview

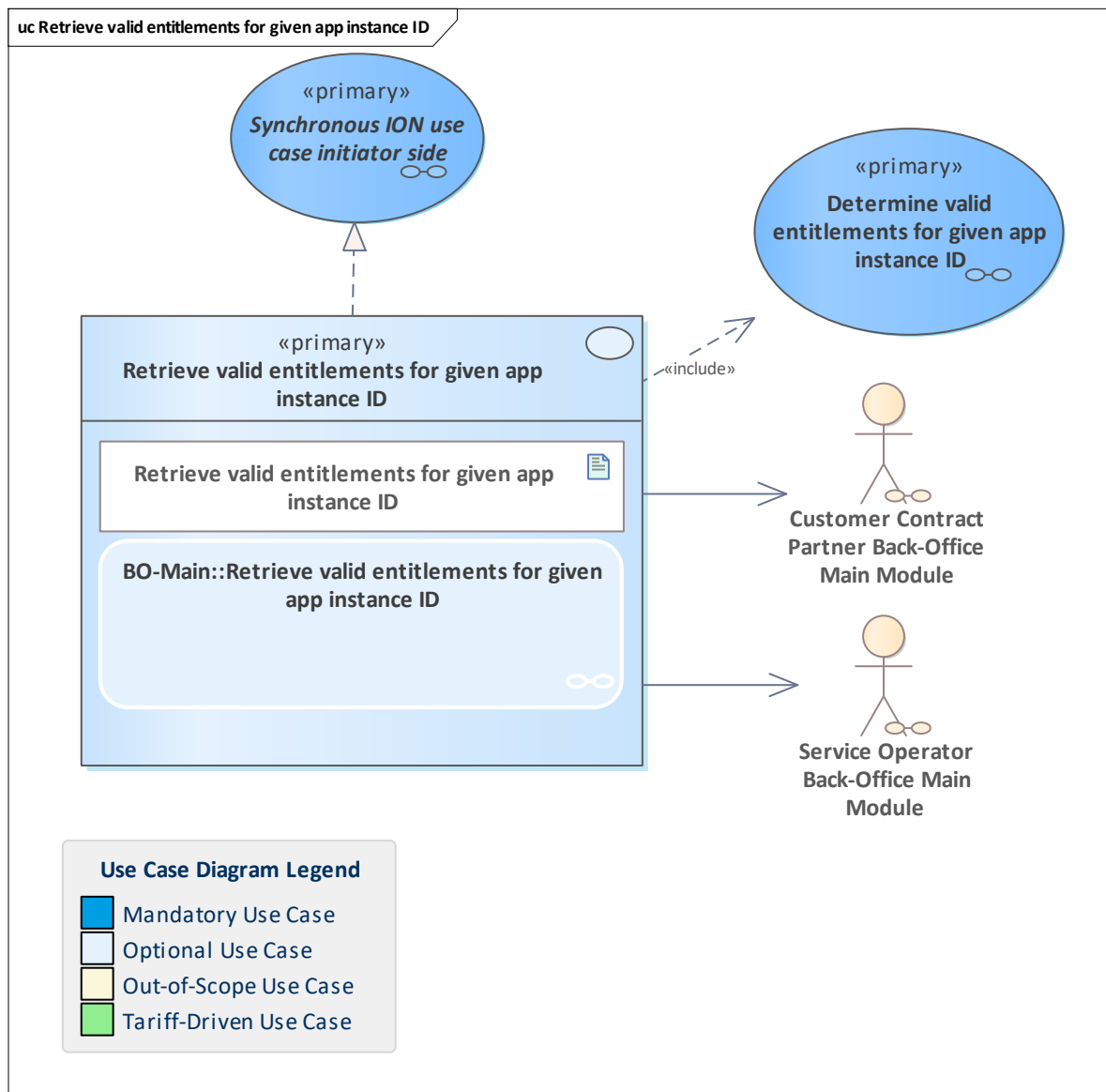
Optional: Retrieve valid entitlements for given app instance ID

Optional: Trigger entitlement issuance

Optional: Handle entitlement validated notification from contractual perspective

### 23.2 Use Cases

#### 23.2.1 **Optional:** Retrieve valid entitlements for given app instance ID

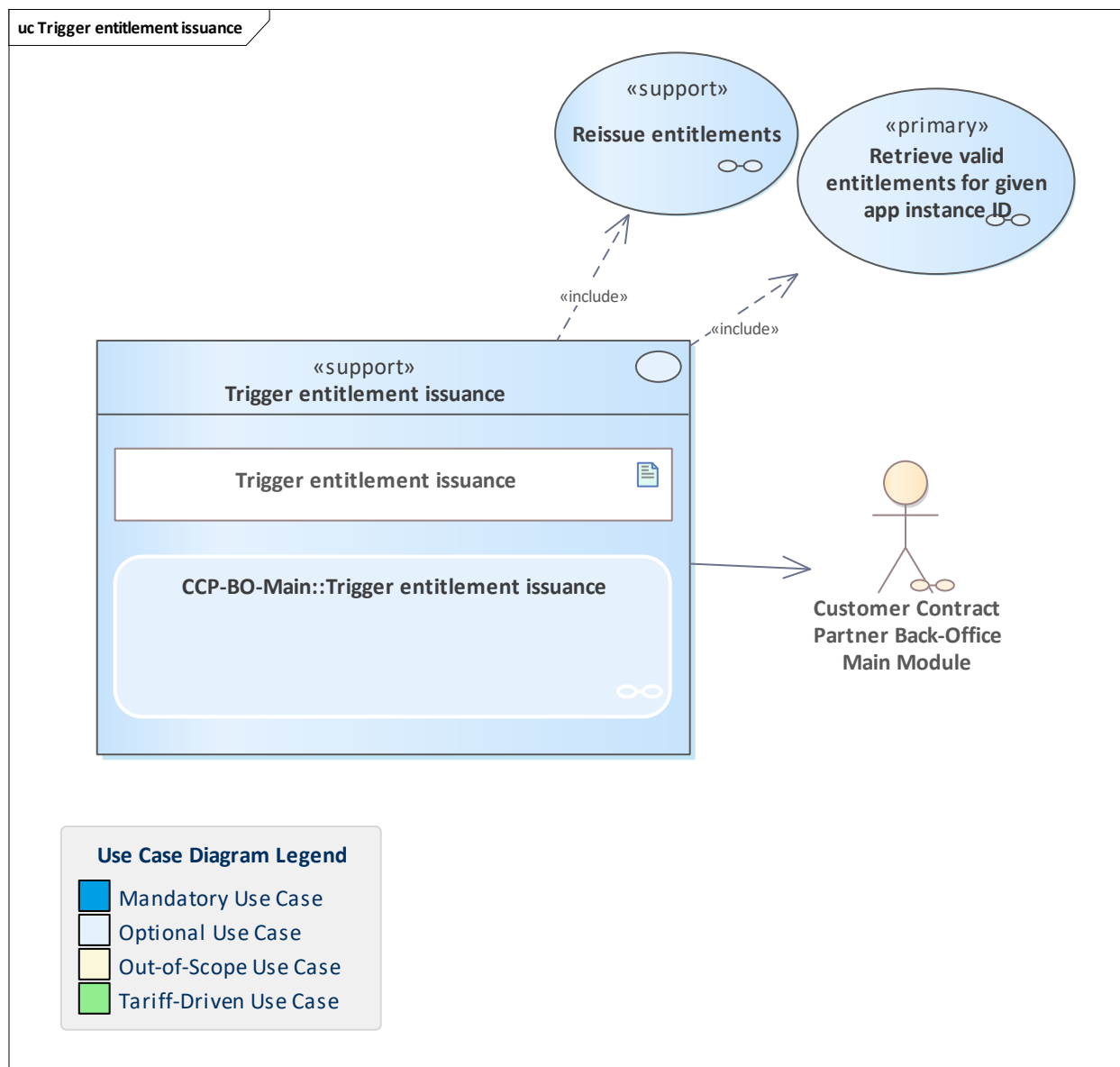


Use Case	<u>Retrieve valid entitlements for given app instance ID</u>
<b>Description</b>	<p>The pCCP of the user medium with an application retrieves information about already issued and valid entitlements from each product owner that might come into question by sending the application instance ID.</p> <p>Please note that this is a snapshot of the entitlements valid at a given timestamp.</p> <p>Only the pCCP is informed about application hotlist entries or even an existing blocking of the application. If this use case is used as part of a user medium/entitlement replacement, the pCCP should be the only entity involved.</p> <p>However, this use case can be used by a SO to access entitlement information inside a penalty fare notice process.</p> <p>It should be noted that neither the requesting SO nor the PO has information about blocked applications. Likewise, the PO has no information about the hotlist of applications. The requesting SO must check the hotlist of applications.</p>
<b>Initiating Actor</b>	



<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a> <a href="#">Service Operator Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Determine valid entitlements for given app instance ID</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Synchronous ION use case initiator side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Valid on : ZonedDate</a> <a href="#">App instance ID : AppInstanceId</a>
<b>Outputs</b>	<a href="#">Complete list of valid entitlements :</a> <a href="#">EntitlementOfGivenUserMedium</a>
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">BO-Main::Retrieve valid entitlements for given app instance ID</a>

## 23.2.2 Optional: Trigger entitlement issuance

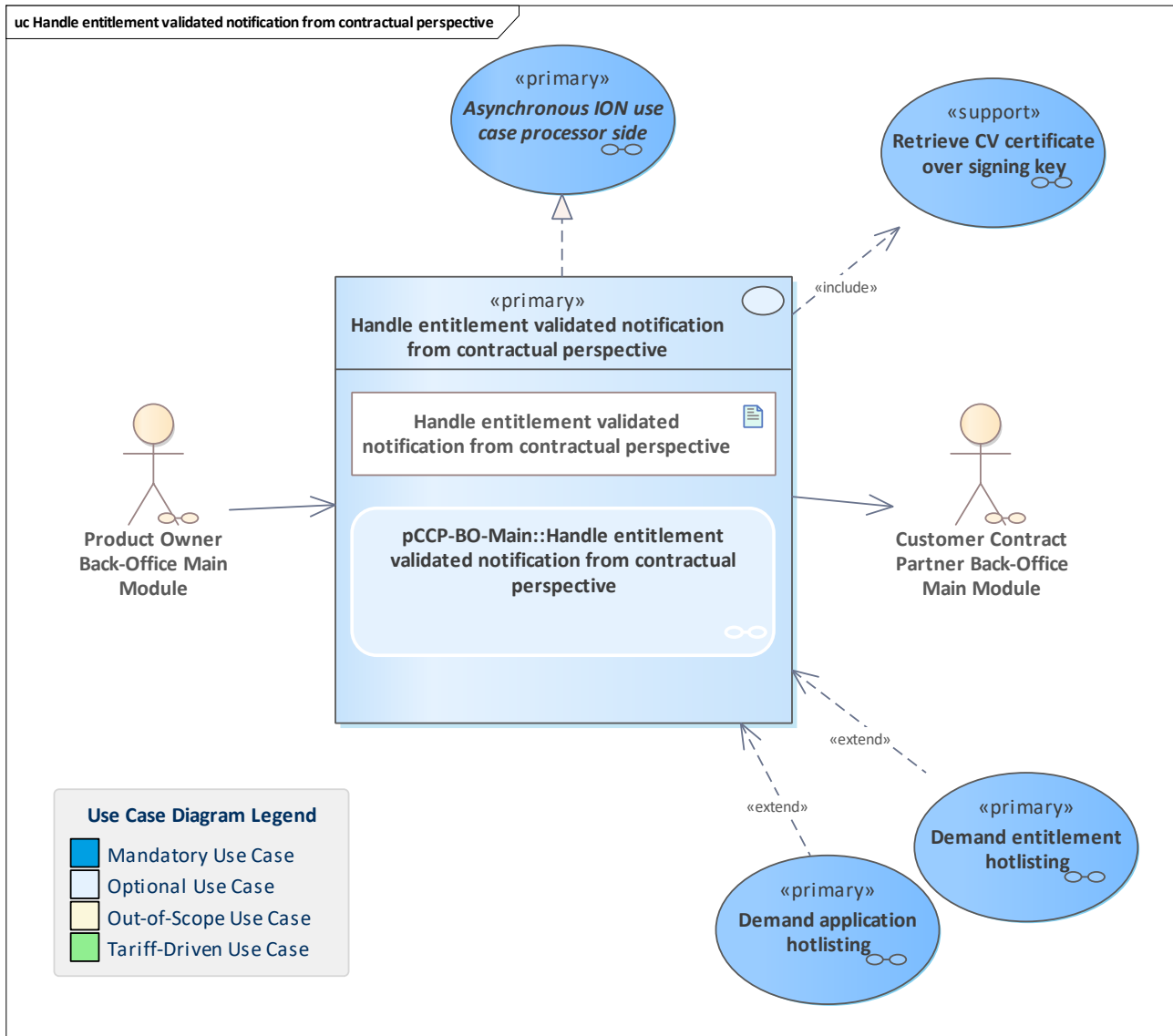


<b>Use Case</b>	<a href="#">Trigger entitlement issuance</a>
<b>Description</b>	A customer needs a new user medium with his entitlements which were located on the old one. Information regarding these entitlements is gathered. New entitlements are issued based on that information.
<b>Initiating Actor</b>	
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	
<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve valid entitlements for given app instance ID</a> / <a href="#">Reissue entitlements</a>
<b>Linked Use Cases</b>	



<b>(Realises)</b>	
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Valid on : ZonedDate</a> <a href="#">CCP : OrganisationId</a> <a href="#">App instance ID of old user medium : AppInstanceId</a> <a href="#">App instance ID of new user medium : AppInstanceId</a>
<b>Outputs</b>	
<b>Error Cases</b>	
<b>Activity Diagram</b>	<a href="#">CCP-BO-Main::Trigger entitlement issuance</a>

## 23.2.3 Optional: Handle entitlement validated notification from contractual perspective



<b>Use Case</b>	<a href="#">Handle entitlement validated notification from contractual perspective</a>
<b>Description</b>	Handle an entitlement validated notification from the contractual perspective. The entitlement validated notification is received by the pCCP. The pCCP does its checks and monitoring from the contractual perspective regarding the correct execution of the validation. In this context, the signature of the embedded attestation is verified.
<b>Initiating Actor</b>	<a href="#">Product Owner Back-Office Main Module</a>
<b>Reacting Actor</b>	<a href="#">Customer Contract Partner Back-Office Main Module</a>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Linked Use Cases (Extended By)</b>	<a href="#">Demand application hotlisting</a> / <a href="#">Demand entitlement hotlisting</a>



<b>Linked Use Cases (Includes)</b>	<a href="#">Retrieve CV certificate over signing key</a>
<b>Linked Use Cases (Realises)</b>	<a href="#">Asynchronous ION use case processor side</a>
<b>Base Activity</b>	
<b>Inputs</b>	<a href="#">Forward entitlement validated notification :</a> <a href="#">forwardEntitlementValidatedNotification</a>
<b>Outputs</b>	<a href="#">Forward entitlement validated notification response :</a> <a href="#">forwardEntitlementValidatedNotificationResponse</a>
<b>Error Cases</b>	<a href="#">E CCP RECEIVER IS NOT ENTITY OWNER</a> <a href="#">E CO BINARY STRUCTURE CANNOT BE PROCESSED</a> <a href="#">E CO WRONG ATTESTATION IN NOTIFICATION</a> <a href="#">E CO WRONG SECURITY LEVEL</a> <a href="#">E CO DUPLICATE UM ATTESTATION</a> <a href="#">Forward entitlement validated notification exception :</a> <a href="#">forwardEntitlementValidatedNotificationException</a>
<b>Activity Diagram</b>	<a href="#">pCCP-BO-Main::Handle entitlement validated notification from contractual perspective</a>



